Voronoi Diagrams and Delaunay Triangulations

Sándor Kisfaludi-Bak

Computaional Geometry
Summer semester 2020

max planck institut
informatik

Overview

- Voronoi diagrams – definition and properties

Overview

- Voronoi diagrams – definition and properties

- Fortune's algorithm (1987)

Overview

- Voronoi diagrams – definition and properties
- Fortune's algorithm (1987)
- Delaunay graphs and triangulations

Overview

- Voronoi diagrams – definition and properties
- Fortune's algorithm (1987)
- Delaunay graphs and triangulations
- Delaunay triangulation via divide and conquer (Guibas and Stolfi, 1985)

Overview

- Voronoi diagrams – definition and properties

- Fortune's algorithm (1987)

- Delaunay graphs and triangulations

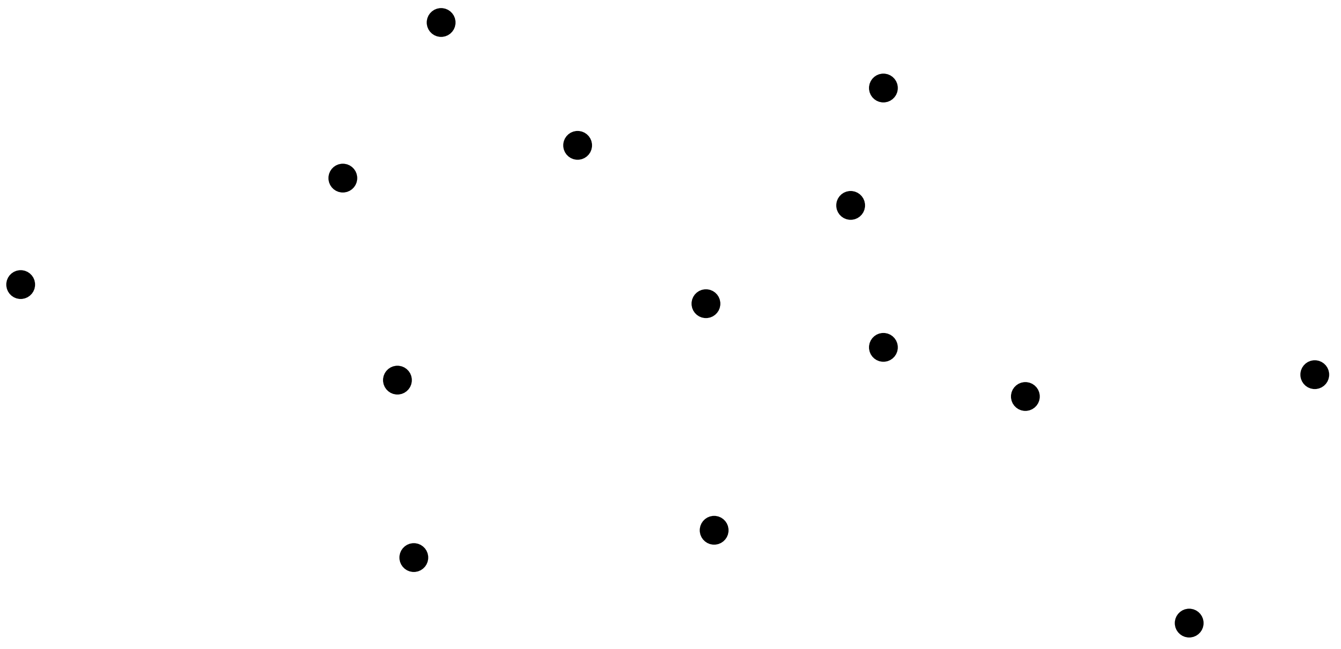
- Delaunay triangulation via divide and conquer (Guibas and Stolfi, 1985)

- Lifting to a paraboloid; computation via convex hull

Next lecture!

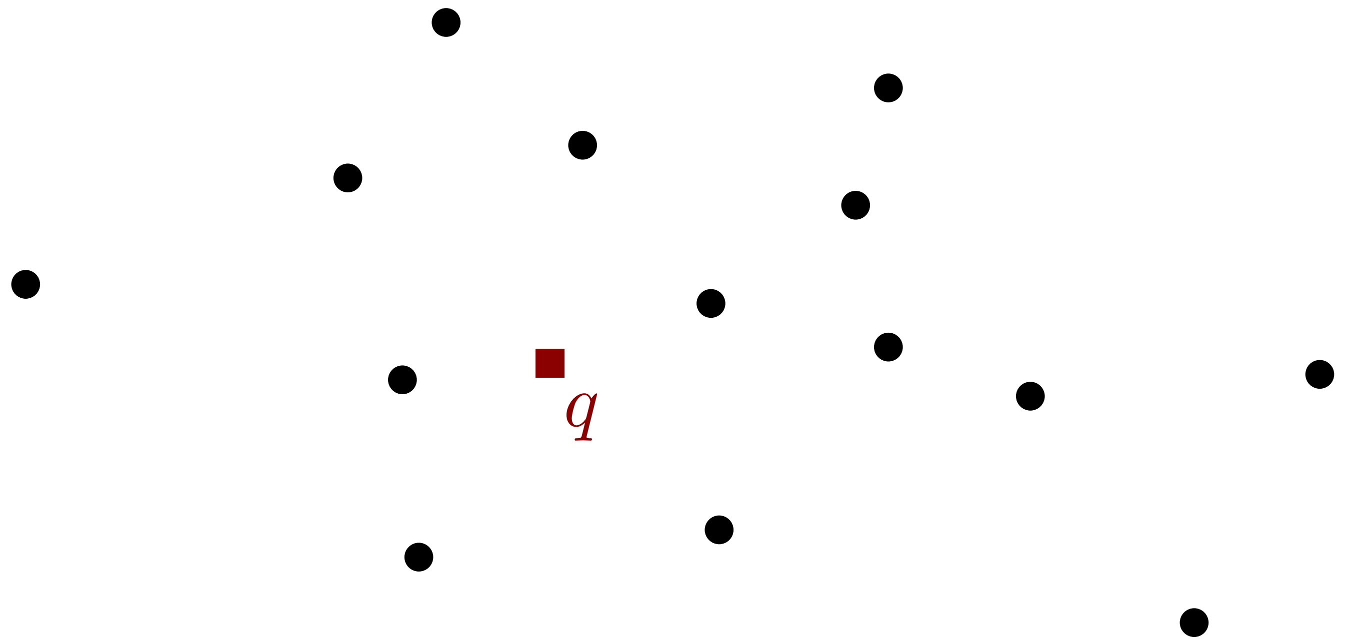
Motivation – nearest neighbor

Given: $P \subset \mathbb{R}^2$.



Motivation – nearest neighbor

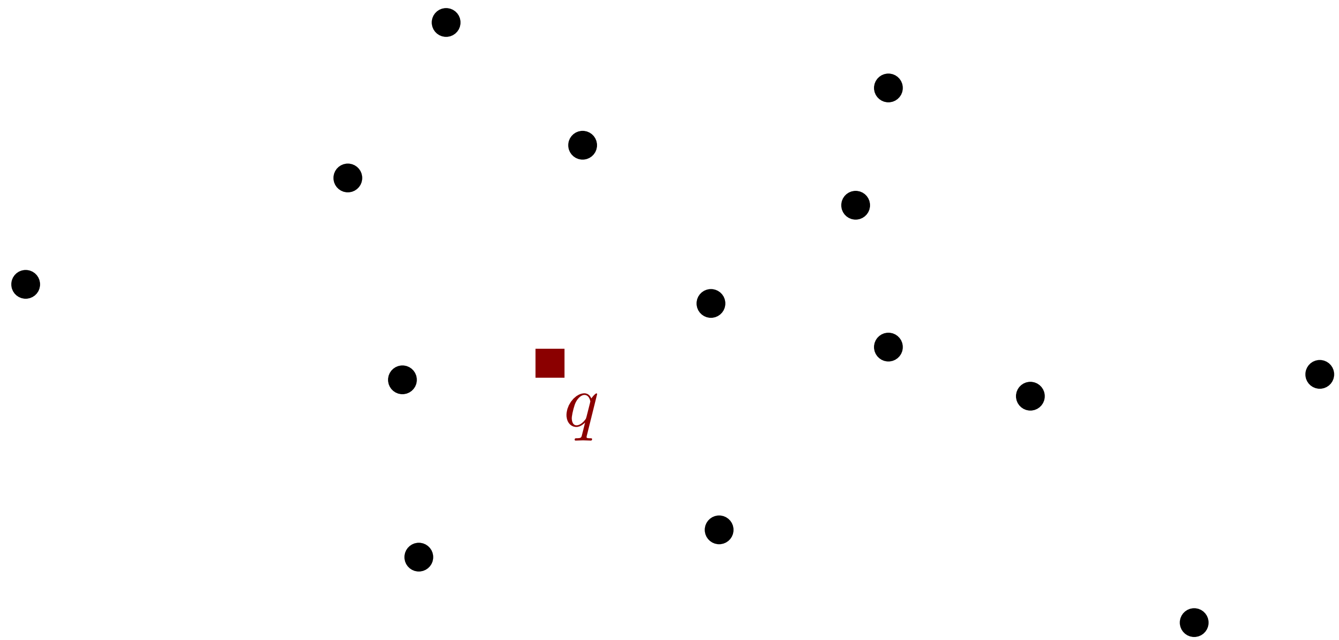
Given: $P \subset \mathbb{R}^2$.



What is the nearest point in P to a given query point $q \in \mathbb{R}^2$?

Motivation – nearest neighbor

Given: $P \subset \mathbb{R}^2$.



What is the nearest point in P to a given query point $q \in \mathbb{R}^2$?

- Where in P should I get my ice cream if I'm at q?
- Accident at q. Which hospital in P should send helicopter?

Voronoi diagram

The Voronoi diagram of $P \subset \mathbb{R}^d$ is the partition of $\mathbb{R}^d$ according to the closest point of P.

Voronoi diagram

The Voronoi diagram of $P \subset \mathbb{R}^d$ is the partition of $\mathbb{R}^d$ according to the closest point of P.

If $|P| = n$, then partition into n cells
s.t. cell of $p \in P$ consist of $q \in \mathbb{R}^d$ where

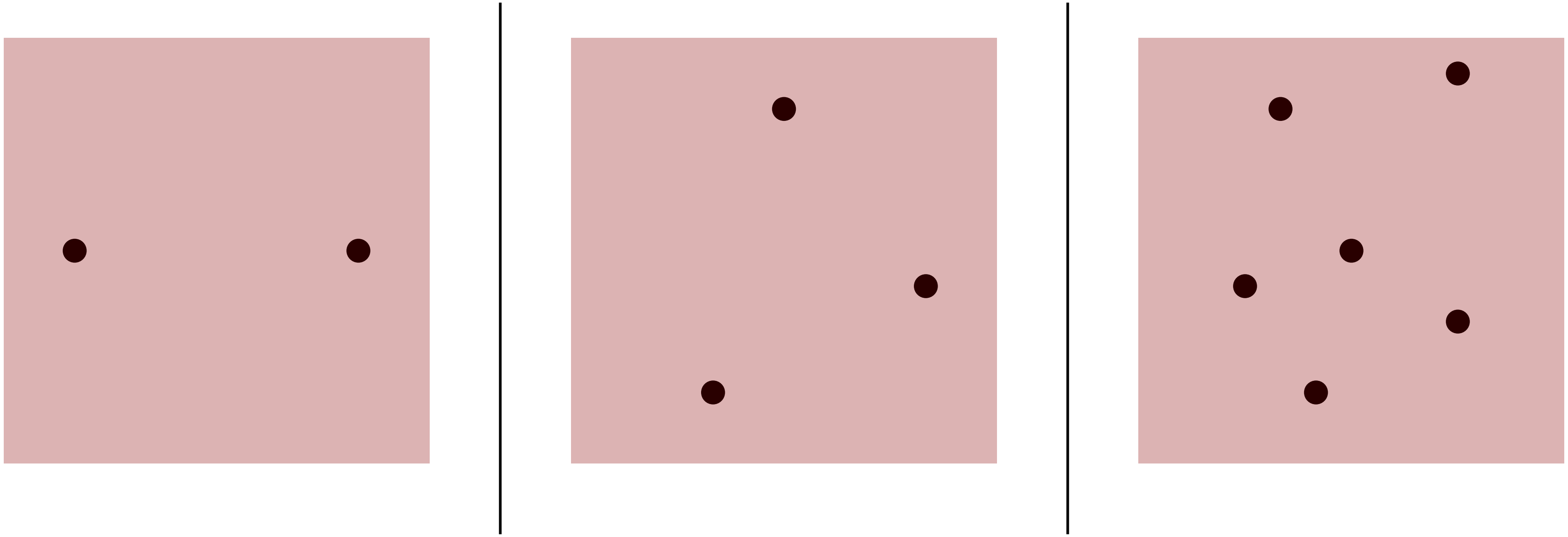
$$\mathrm{dist}(q,p) < \mathrm{dist}(q,p') \text{ for all } p' \in P \setminus \{p\}$$

Voronoi diagram

The Voronoi diagram of $P \subset \mathbb{R}^d$ is the partition of $\mathbb{R}^d$ according to the closest point of P.

If $|P| = n$, then partition into n cells
s.t. cell of $p \in P$ consist of $q \in \mathbb{R}^d$ where

$$\operatorname{dist}(q, p) < \operatorname{dist}(q, p') \text{ for all } p' \in P \setminus \{p\}$$

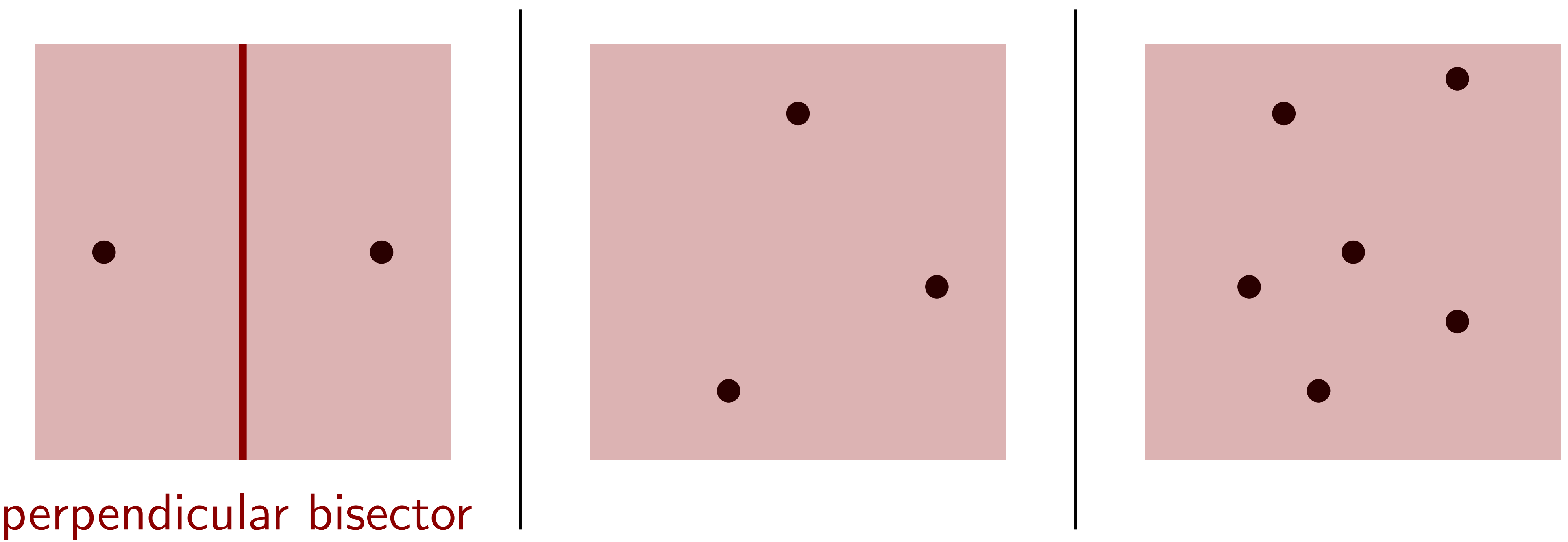


Voronoi diagram

The Voronoi diagram of $P \subset \mathbb{R}^d$ is the partition of $\mathbb{R}^d$ according to the closest point of P.

If $|P| = n$, then partition into n cells s.t. cell of $p \in P$ consist of $q \in \mathbb{R}^d$ where

$$\mathrm{dist}(q, p) < \mathrm{dist}(q, p') \text{ for all } p' \in P \setminus \{p\}$$

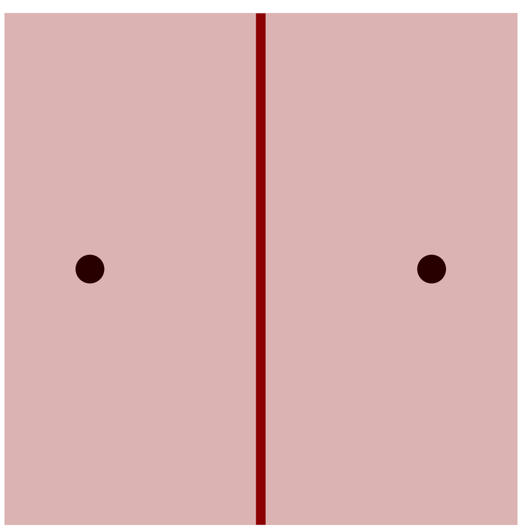


Voronoi diagram

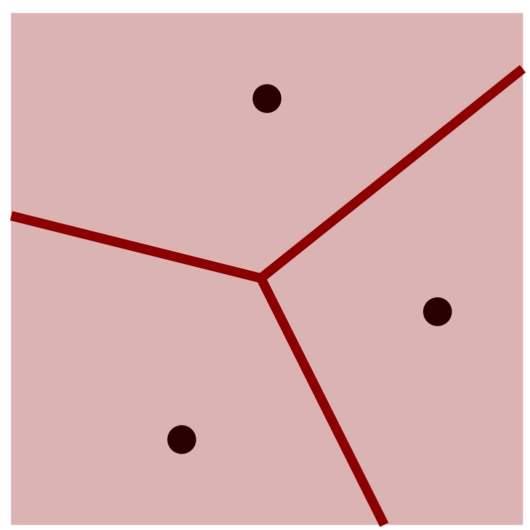
The Voronoi diagram of $P \subset \mathbb{R}^d$ is the partition of $\mathbb{R}^d$ according to the closest point of P.

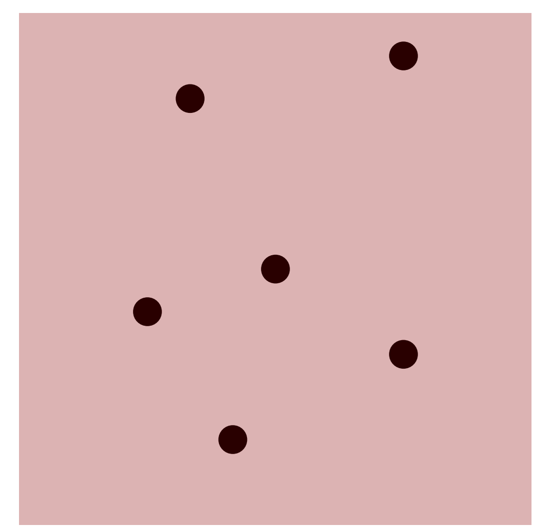
If $|P| = n$, then partition into n cells
s.t. cell of $p \in P$ consist of $q \in \mathbb{R}^d$ where

$$\mathrm{dist}(q, p) < \mathrm{dist}(q, p') \text{ for all } p' \in P \setminus \{p\}$$



perpendicular bisector



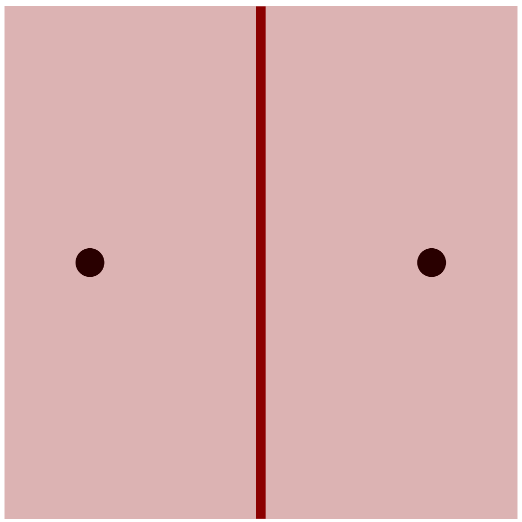


Voronoi diagram

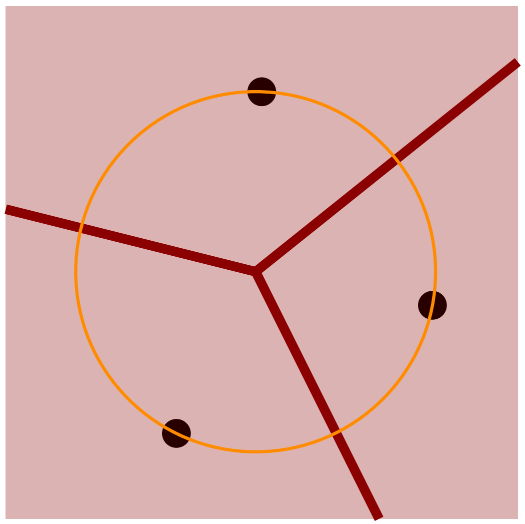
The Voronoi diagram of $P \subset \mathbb{R}^d$ is the partition of $\mathbb{R}^d$ according to the closest point of P.

If $|P| = n$, then partition into n cells
s.t. cell of $p \in P$ consist of $q \in \mathbb{R}^d$ where

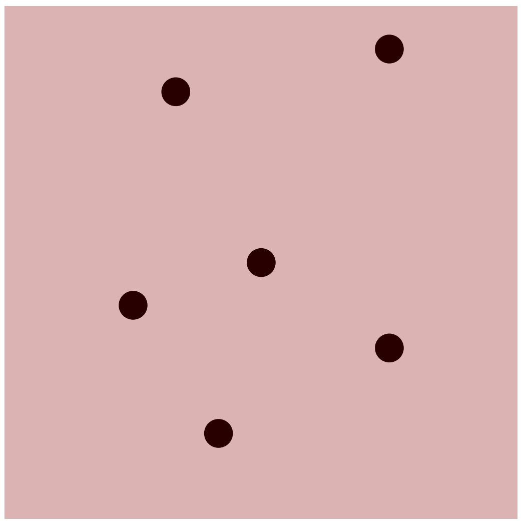
$$\mathrm{dist}(q, p) < \mathrm{dist}(q, p') \text{ for all } p' \in P \setminus \{p\}$$



perpendicular bisector



bisectors, center of circumcircle

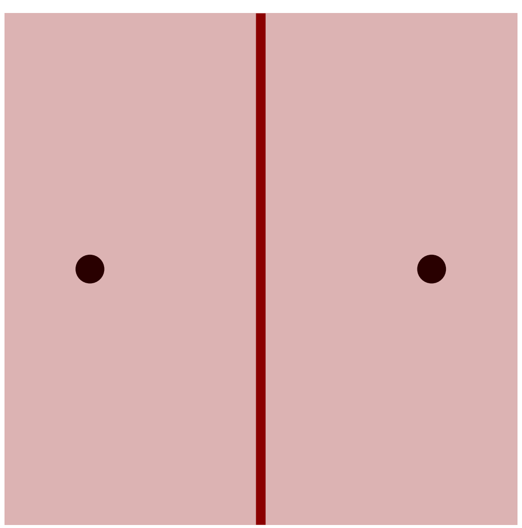


Voronoi diagram

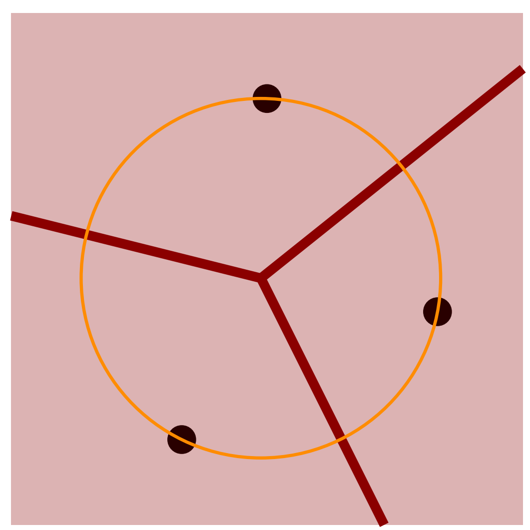
The Voronoi diagram of $P \subset \mathbb{R}^d$ is the partition of $\mathbb{R}^d$ according to the closest point of P.

If $|P| = n$, then partition into n cells
s.t. cell of $p \in P$ consist of $q \in \mathbb{R}^d$ where

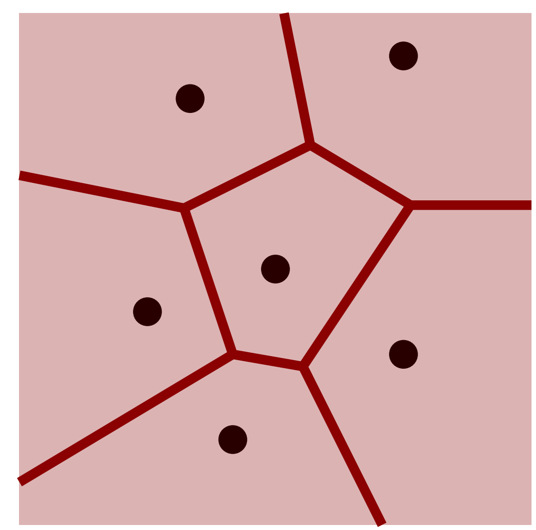
$$\mathrm{dist}(q,p) < \mathrm{dist}(q,p') \text{ for all } p' \in P \setminus \{p\}$$



perpendicular bisector



bisectors, center of circumcircle



Historical notes

Voronoi diagram = Dirichlet tessellation

Goes back to Descartes

Historical notes

Voronoi diagram = Dirichlet tessellation

1907 1850

Goes back to Descartes 1644

Complexity and properties in $\mathbb{R}^2$

Each Cell(p) is intersection of half-planes.
Each cell is convex (bounded or unbounded) polygon.

Vor(P): collection of segments and rays on cell boundaries

If P has 3 non-collinear pts
$\Rightarrow$ Vor(P) is connected

Complexity and properties in $\mathbb{R}^2$

Each Cell(p) is intersection of half-planes.
Each cell is convex (bounded or unbounded) polygon.

Vor(P): collection of segments and rays on cell boundaries

If P has 3 non-collinear pts
$\Rightarrow$ Vor(P) is connected

Lemma Vor(P) has total complexity $O(n)$.

Complexity and properties in $\mathbb{R}^2$

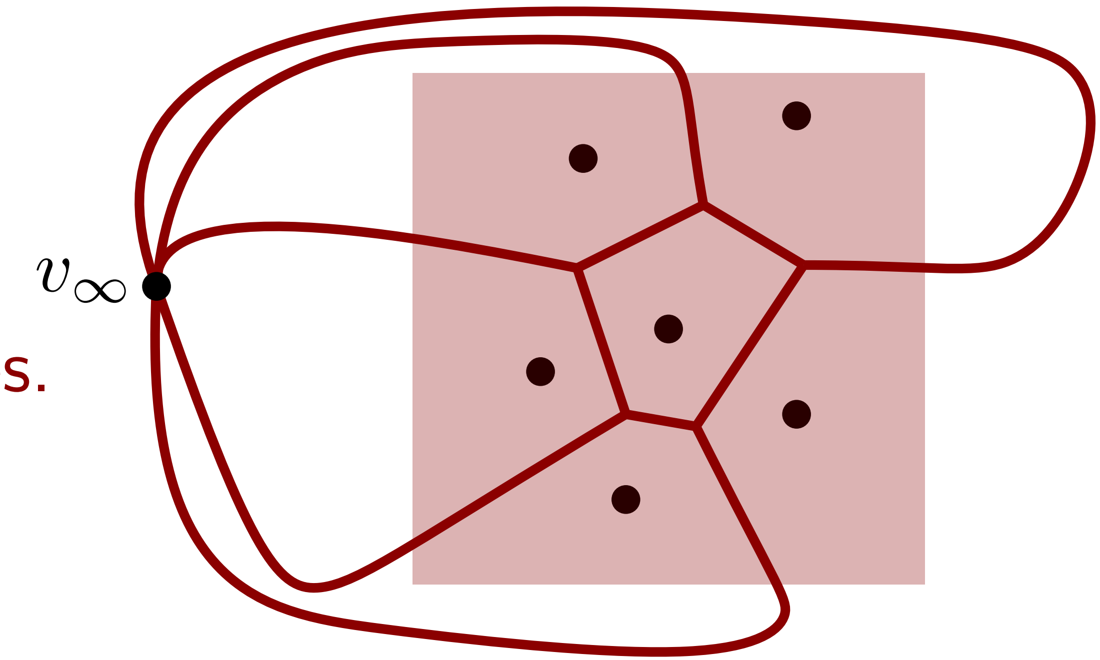
Each Cell(p) is intersection of half-planes.
Each cell is convex (bounded or unbounded) polygon.

Vor(P): collection of segments and rays on cell boundaries

If P has 3 non-collinear pts
$\Rightarrow$ Vor(P) is connected

Lemma Vor(P) has total complexity $O(n)$.

Proof. There are n cells.
Euler's formula
$\Rightarrow O(n)$ edges, $O(n)$ vertices.



Circumcircles in Voronoi diagrams

$C(q)$: largest circle around q whose interior has no pts from P

Lemma

(i) q is a vertex of Vor(P) iff $C(q)$ has at least 3 points of P

(ii) q is on edge btw. cell(p) and cell(p') iff $C(q) \cap P = \{p, p'\}$

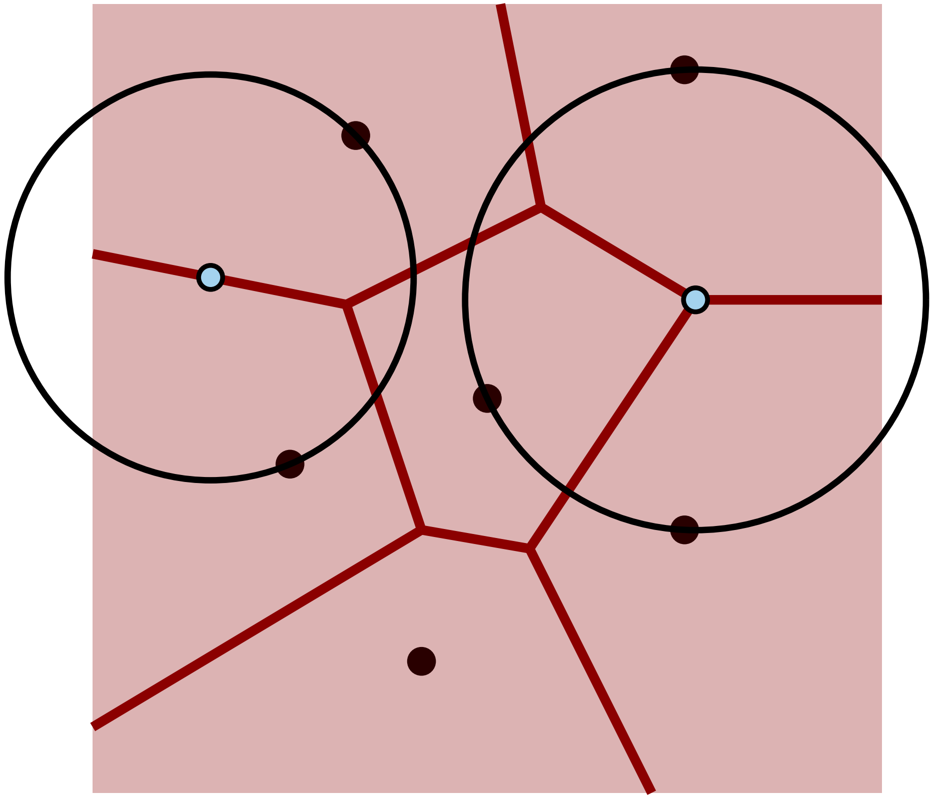
Circumcircles in Voronoi diagrams

$C(q)$: largest circle around q whose interior has no pts from P

Lemma

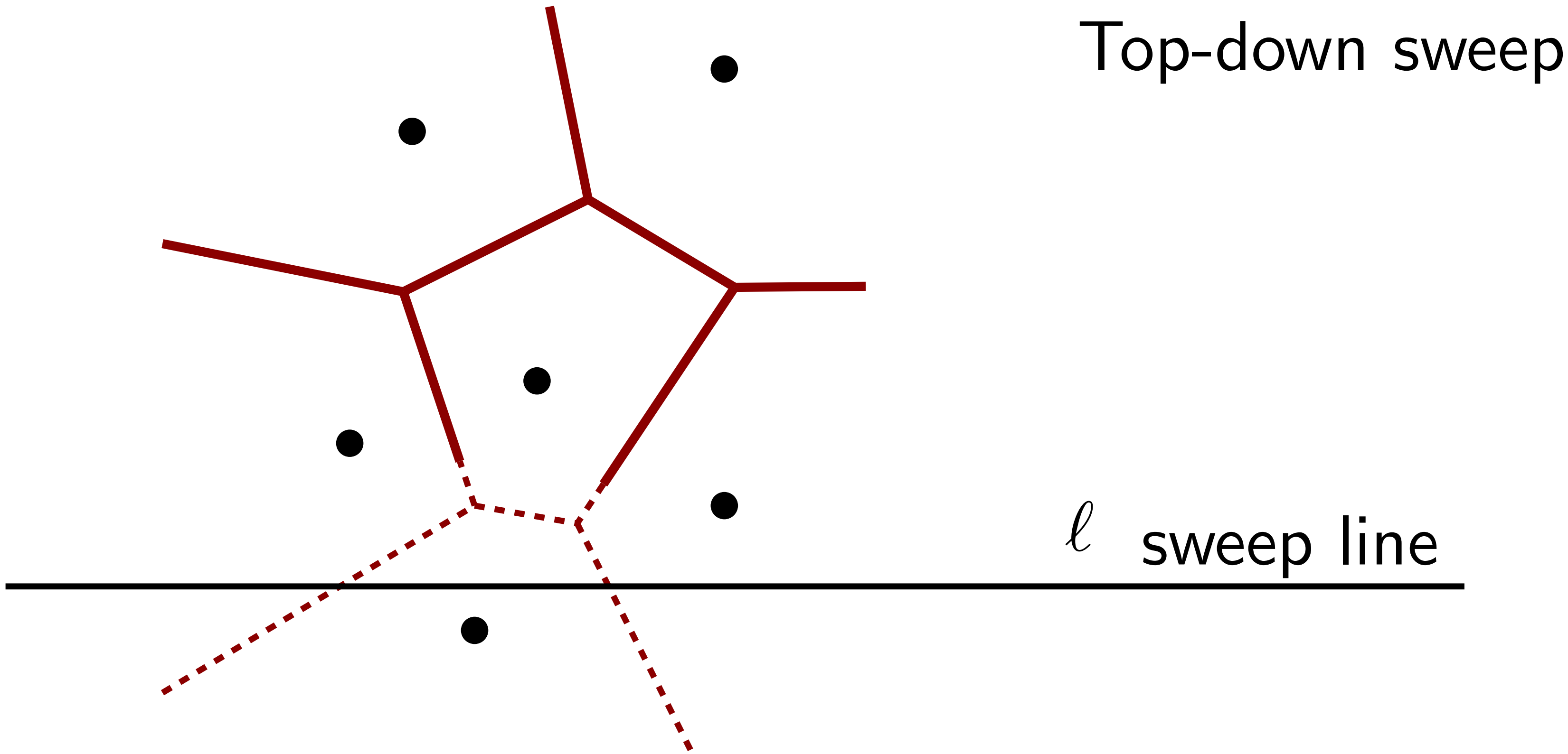
(i) q is a vertex of Vor(P) iff $C(q)$ has at least 3 points of P

(ii) q is on edge btw. cell(p) and cell(p') iff $C(q) \cap P = \{p, p'\}$

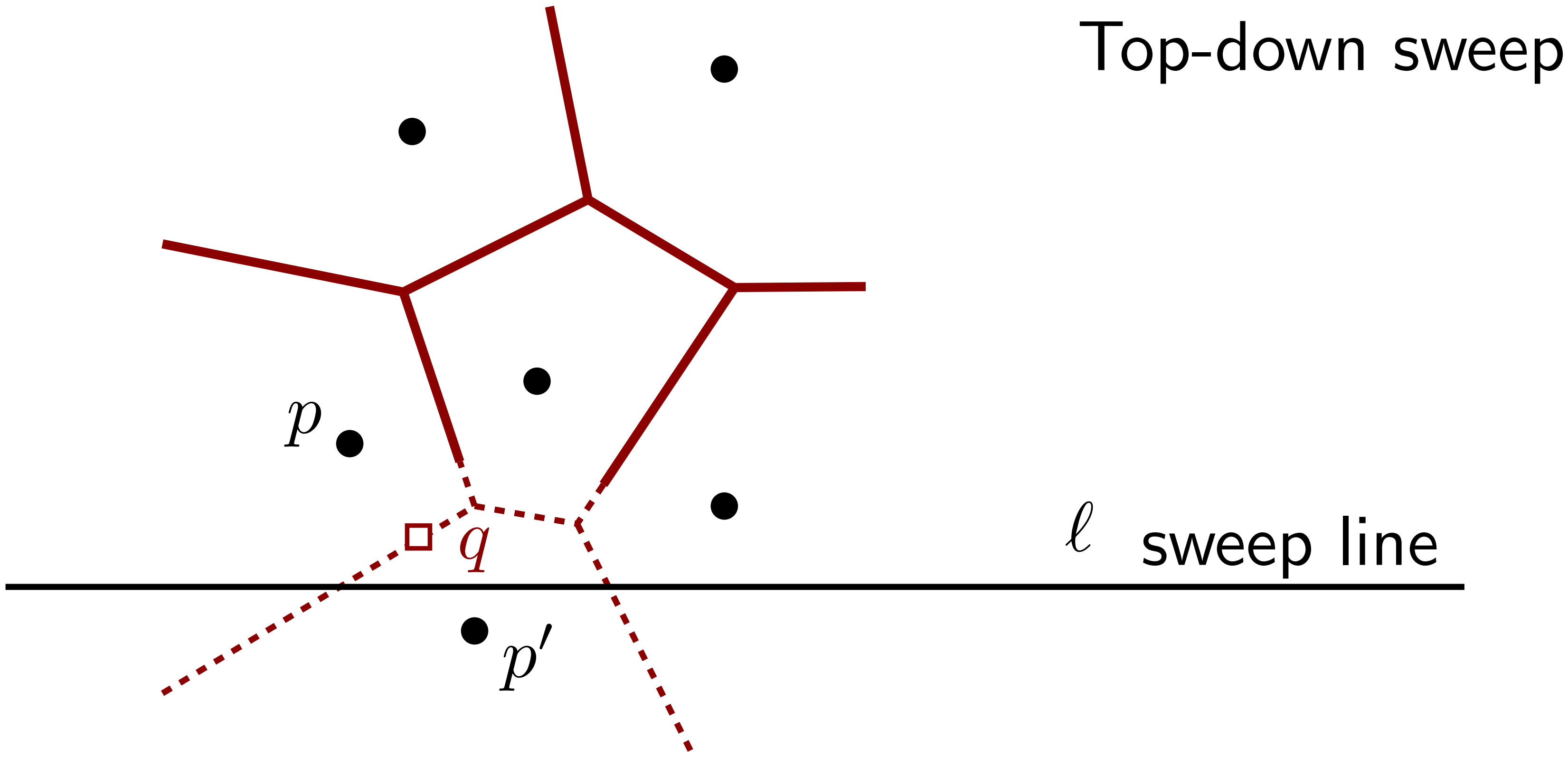


Fortune's algorithm (1987)

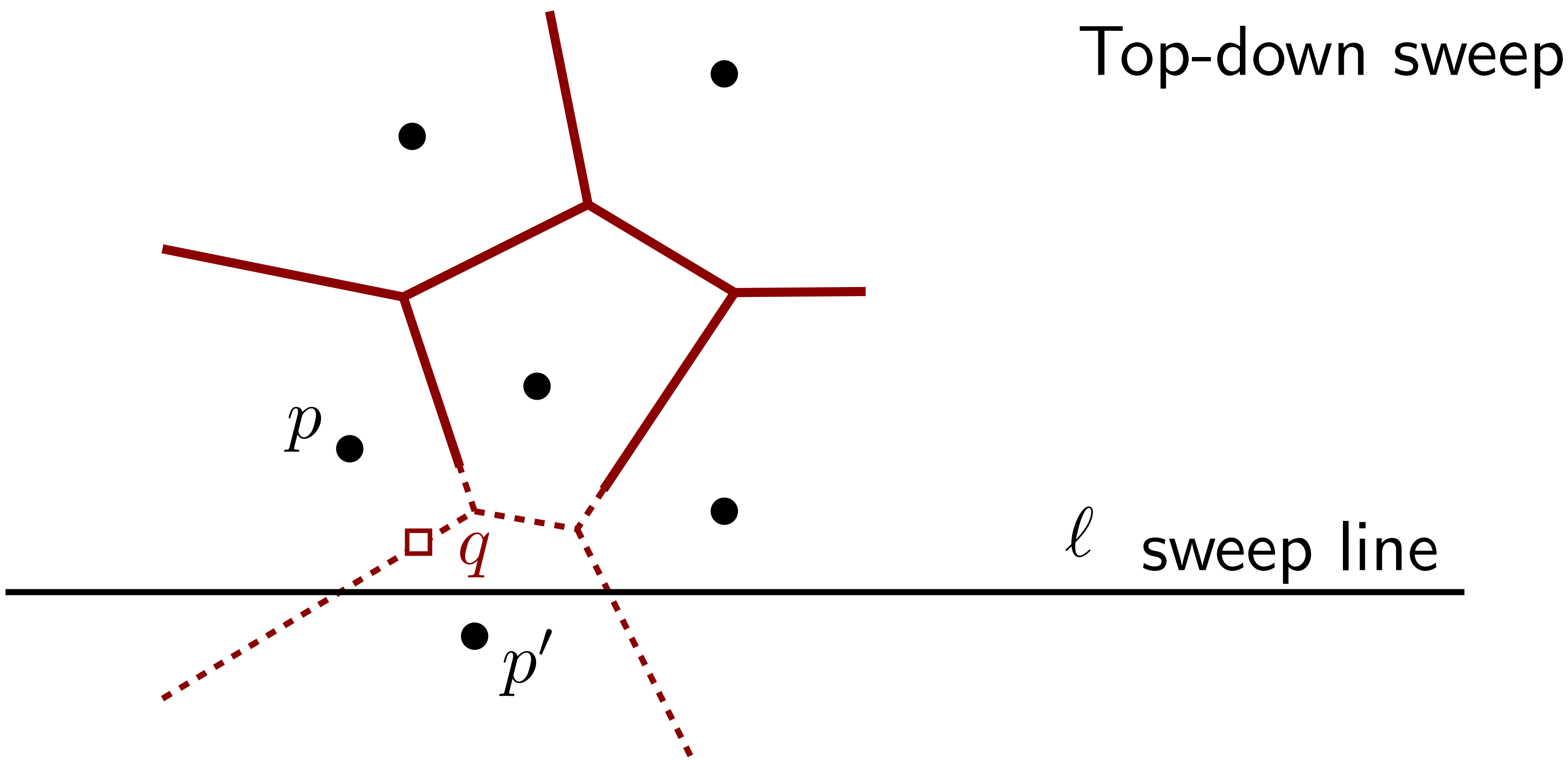
Sweeping with a wavefront



Sweeping with a wavefront



Sweeping with a wavefront

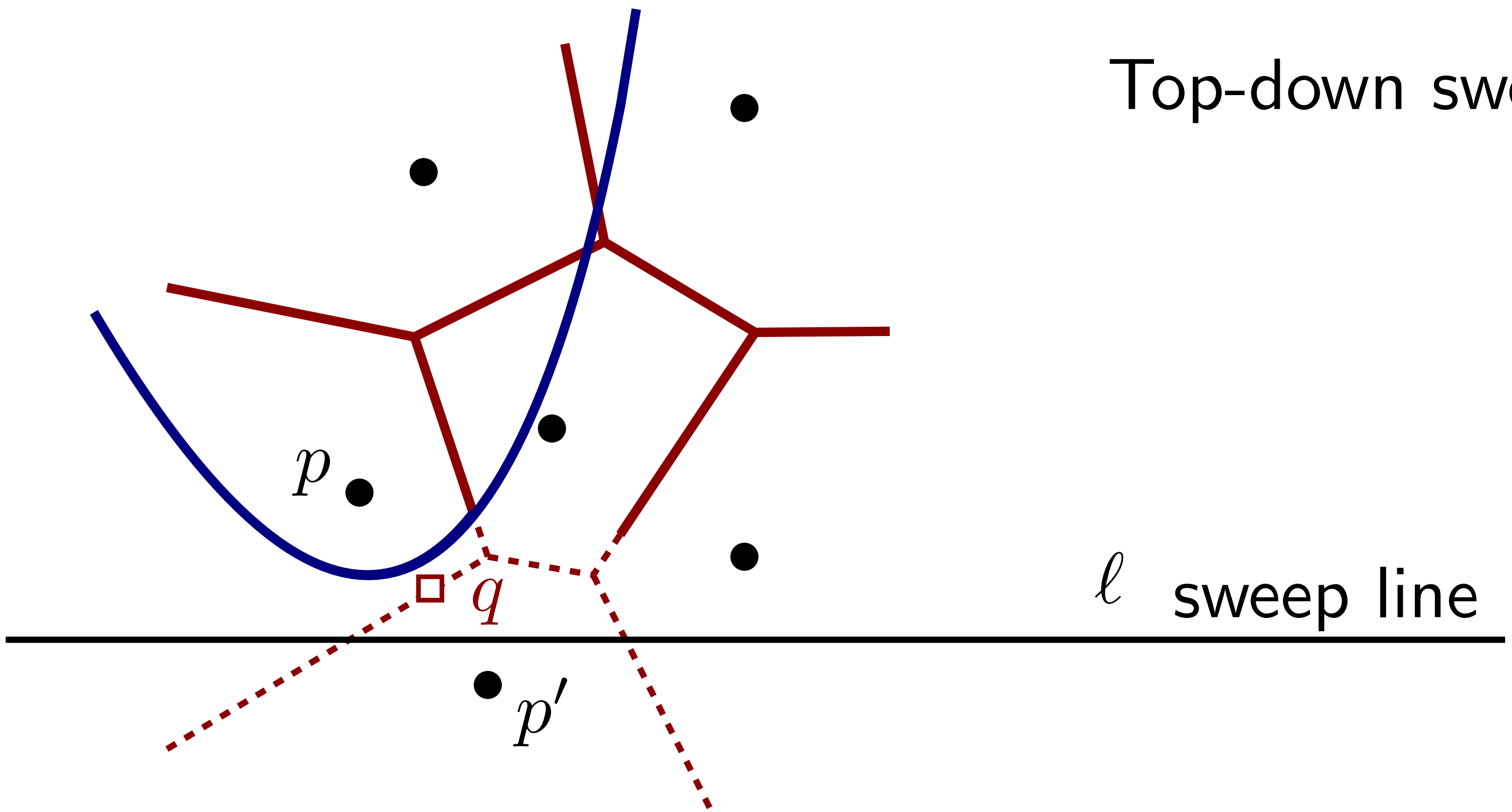


If q is on undiscovered edge btw. Cell(p) and Cell(p')

$$\mathrm{dist}(p,q) = \mathrm{dist}(p',q) \Rightarrow \mathrm{dist}(p,q) \geq \mathrm{dist}(q,\ell)$$

q is below the parabola with focus p and axis ℓ

Sweeping with a wavefront

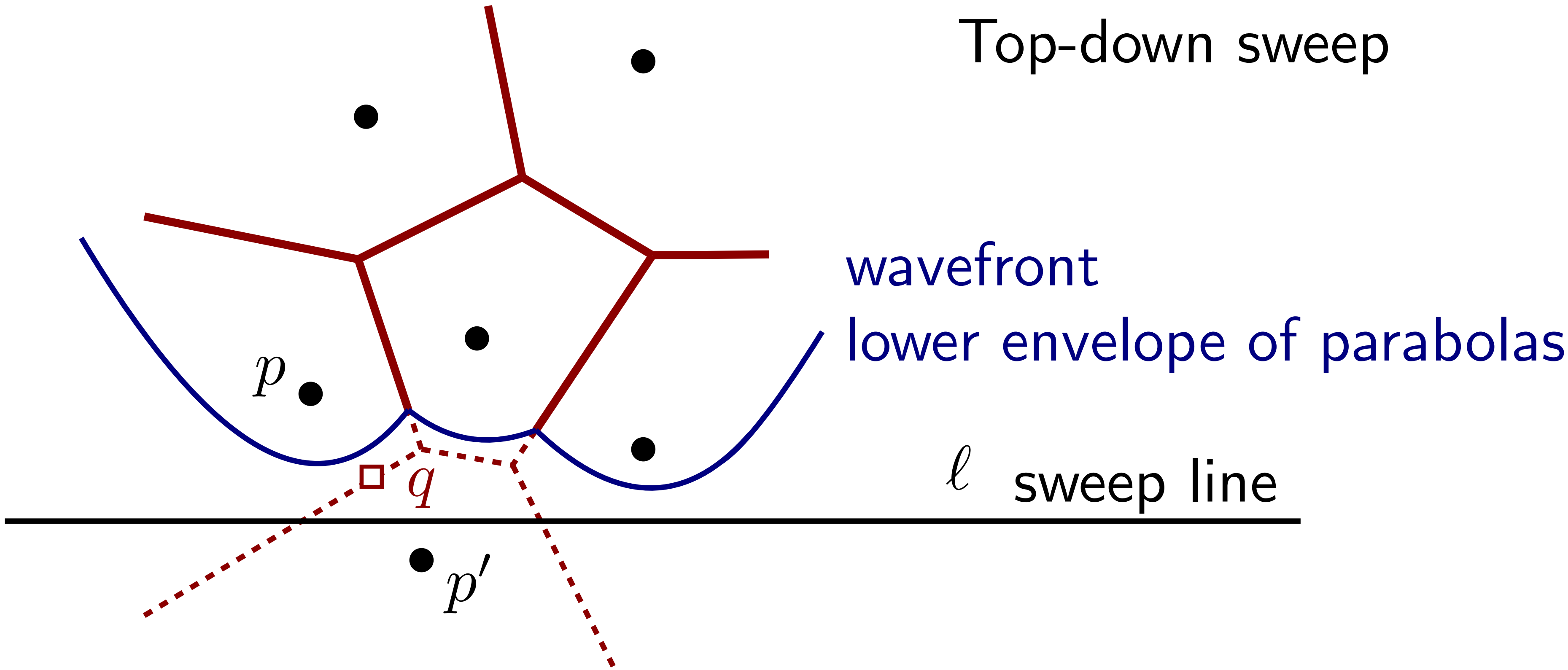


If q is on undiscovered edge btw. Cell(p) and Cell(p')

$$\mathrm{dist}(p,q) = \mathrm{dist}(p',q) \Rightarrow \mathrm{dist}(p,q) \geq \mathrm{dist}(q,\ell)$$

q is below the parabola with focus p and axis ℓ

Sweeping with a wavefront



If q is on undiscovered edge btw. Cell(p) and Cell(p')

$$\mathrm{dist}(p,q) = \mathrm{dist}(p',q) \Rightarrow \mathrm{dist}(p,q) \geq \mathrm{dist}(q,\ell)$$

q is below the parabola with focus p and axis ℓ

Vor(P) above waverfront is correct

Sweeping

- Invariant
  Part of diagram above wavefront is correctly computed

Sweeping

- Invariant
  Part of diagram above wavefront is correctly computed
- Sweep line structure
  Intersection of diagram with ℓ

Sweeping

- Invariant
  Part of diagram above wavefront is correctly computed

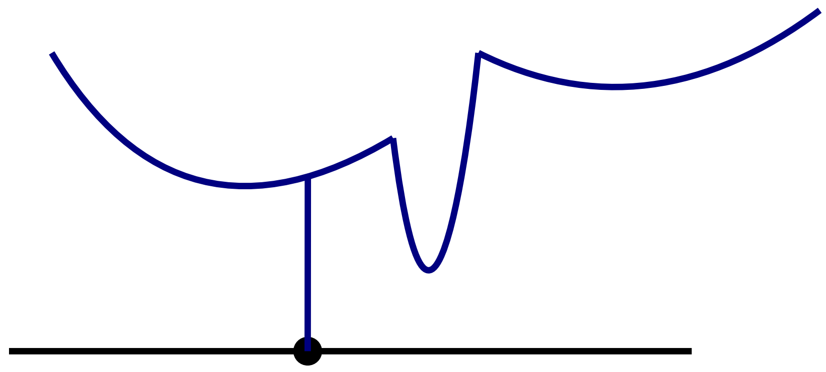
- Sweep line structure
  ~~Intersection of diagram with ℓ~~
  Wavefront (vertices and parabolas in order)

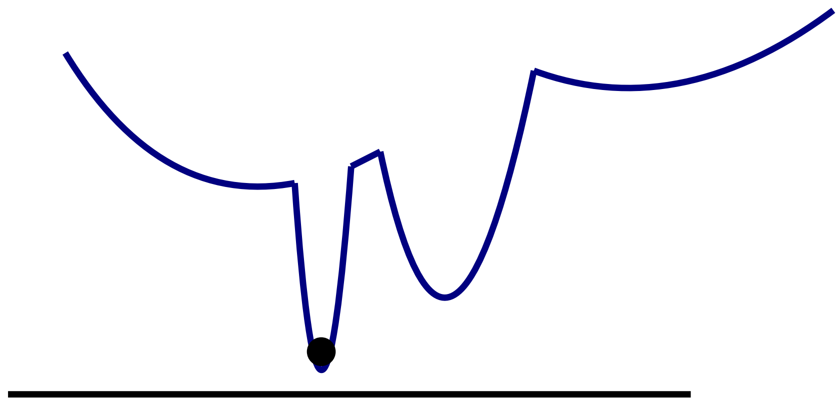
Sweeping

- Invariant
  Part of diagram above wavefront is correctly computed
- Sweep line structure
  ~~Intersection of diagram with ℓ~~
  Wavefront (vertices and parabolas in order)
- Event queue:
  new parabola on wavefront
  remove arc from wavefront

Sweeping

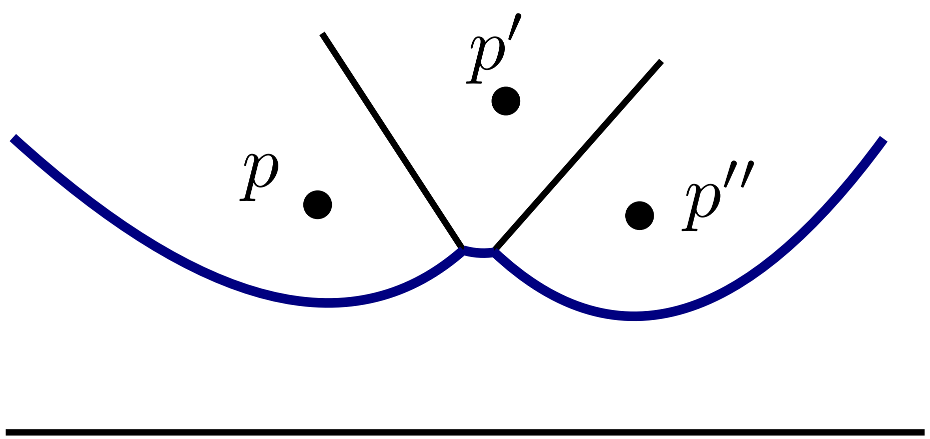
- Invariant
  Part of diagram above wavefront is correctly computed
- Sweep line structure
  ~~Intersection of diagram with ℓ~~
  Wavefront (vertices and parabolas in order)
- Event queue:
  new parabola on wavefront $\Leftrightarrow$ ℓ passes through $p \in P$
  remove arc from wavefront

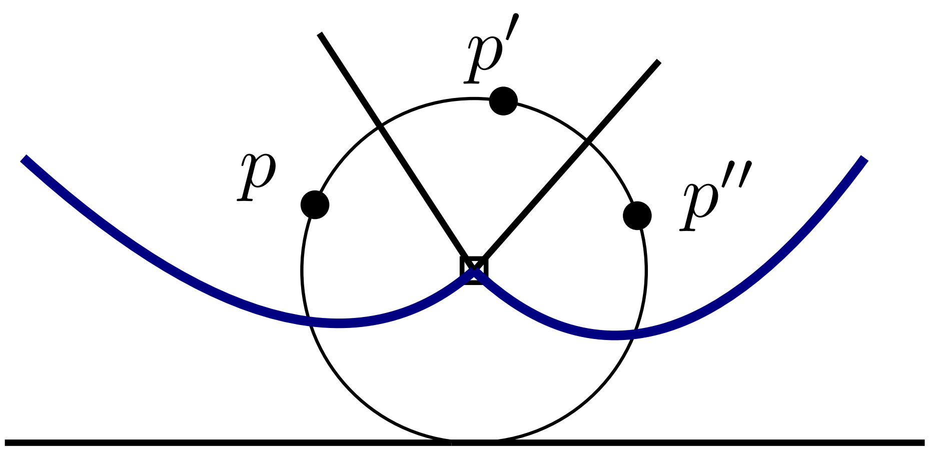




Sweeping

- Invariant
  Part of diagram above wavefront is correctly computed
- Sweep line structure
  ~~Intersection of diagram with ℓ~~
  Wavefront (vertices and parabolas in order)
- Event queue:
  new parabola on wavefront $\Leftrightarrow$ ℓ passes through $p \in P$
  remove arc from wavefront $\Leftrightarrow$ ℓ touches circle $pp'p''$





Wavefront complexity and queue maintenance

Observation. The wavefront consists of at most $2n - 1$ parabolic arcs.

Proof: n new arcs added, each splits an existing arc into at most 2 arcs

Wavefront complexity and queue maintenance

Observation. The wavefront consists of at most $2n - 1$ parabolic arcs.

Proof: n new arcs added, each splits an existing arc into at most 2 arcs

---

Event queue: contains unswept points and some circles $pp'p''$ currently intersected by ℓ

iff parabolas of p, p', p'' are consecutive on wavefront

Wavefront complexity and queue maintenance

Observation. The wavefront consists of at most $2n - 1$ parabolic arcs.

Proof: n new arcs added, each splits an existing arc into at most 2 arcs

---

Event queue: contains unswept points and some circles $pp'p''$ currently intersected by ℓ

iff parabolas of p, p', p'' are consecutive on wavefront

updated with each change to wavefront.

Fortune's sweep more precisely

- Invariant
  Part of diagram above wavefront is correctly computed
  EQ contains:
  - unswept points
  - parabola disappearance events for consecutive arc triplets of wavefront with intersecting circumcircle

Fortune's sweep more precisely

- Invariant
  Part of diagram above wavefront is correctly computed
  EQ contains:
  - unswept points
  - parabola disappearance events for consecutive arc triplets of wavefront with intersecting circumcircle
- Sweep line structure
  Wavefront as self-balancing BST on wavefront vertices, represented by focus pairs (p, p'), ordered left to right

Fortune's sweep more precisely

- Invariant
  Part of diagram above wavefront is correctly computed
  EQ contains:
  - unswept points
  - parabola disappearance events for consecutive arc triplets of wavefront with intersecting circumcircle
- Sweep line structure
  Wavefront as self-balancing BST on wavefront vertices, represented by focus pairs (p, p'), ordered left to right
- Event queue:
  new parabola on wavefront (new point swept)
  remove existing arc from wavefront
  Stored as priority queue

Fortune's sweep more precisely

- Invant
  Part of diagram above wavefront is correctly computed
  EQ contains:
  - unswept points
  - parabola disappearance events for consecutive arc triplets of wavefront with intersecting circumcircle
- Sweep line structure
  Wavefront as self-balancing BST on wavefront vertices, represented by focus pairs (p, p'), ordered left to right
- Event queue:
  new parabola on wavefront (new point swept)
  remove existing arc from wavefront
  Stored as priority queue

$O(n)$ events with $O(\log n)$ time per event $\Rightarrow O(n \log n)$

Fortune's sweep conclusion

Theorem The Voronoi diagram of n points in $\mathbb{R}^2$ can be computed in $O(n \log n)$ time and $O(n)$ space.

Fortune's sweep conclusion

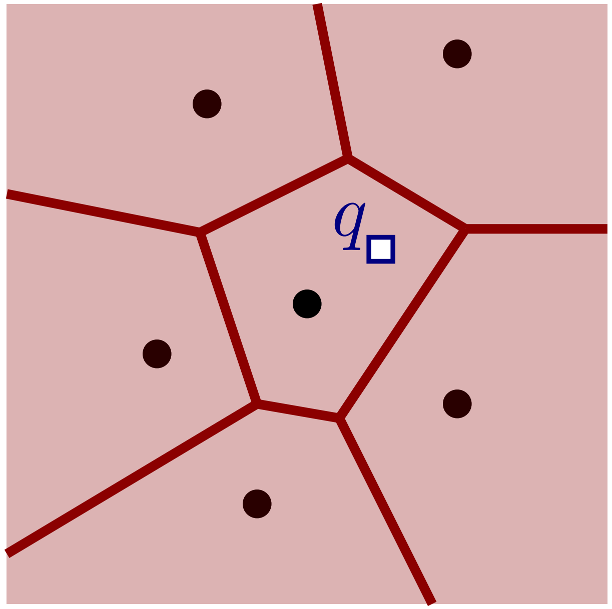
Theorem The Voronoi diagram of n points in $\mathbb{R}^2$ can be computed in $O(n \log n)$ time and $O(n)$ space.

$\Rightarrow$ NEAREST NEIGHBOR solved in $O(n)$ space and $O(\log n)$ query time with a point location data strucutre on the Voronoi diagram.

Fortune's sweep conclusion

Theorem The Voronoi diagram of n points in $\mathbb{R}^2$ can be computed in $O(n \log n)$ time and $O(n)$ space.

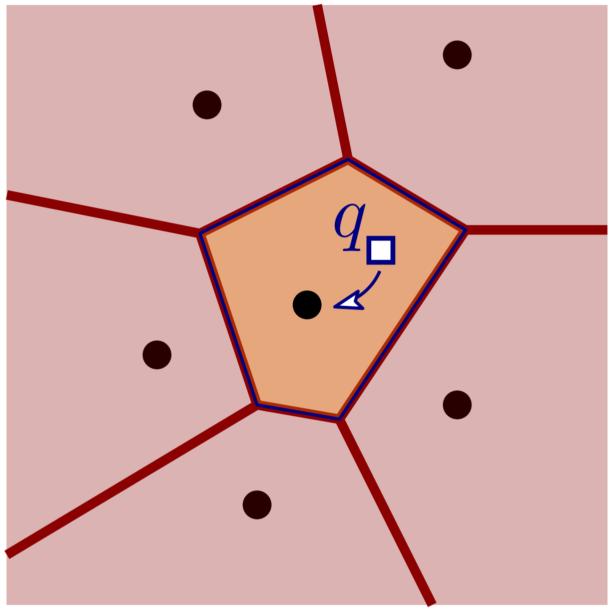
$\Rightarrow$ Nearest Neighbor solved in $O(n)$ space and $O(\log n)$ query time with a point location data strucutre on the Voronoi diagram.



Fortune's sweep conclusion

Theorem The Voronoi diagram of n points in $\mathbb{R}^2$ can be computed in $O(n \log n)$ time and $O(n)$ space.

$\Rightarrow$ NEAREST NEIGHBOR solved in $O(n)$ space and $O(\log n)$ query time with a point location data strucutre on the Voronoi diagram.



General Voronoi diagrams

Voronoi diagram in different metrics:

- Manhattan (L_1), L_p
- Hyperbolic
- Edge weighted planar graph
- abstract (for some definition of "bisector")

General Voronoi diagrams

Voronoi diagram in different metrics:

- Manhattan (L_1), L_p
- Hyperbolic
- Edge weighted planar graph
- abstract (for some definition of "bisector")

Other generalizations:

- of segments
- additively/multiplicatively weighted
- power diagram
- Farthest point
- Order-k

Delaunay triangulations

Triangualtions, complexity

Triangulation of P:

subdivision of $\mathrm{conv}(P)$ into triangles (simplices) whose vertex set is P

Triangualtions, complexity

Triangulation of P:

subdivision of $\mathrm{conv}(P)$ into triangles (simplices) whose vertex set is P

$P \subset \mathbb{R}^2 \Rightarrow$ triangulation has total complexity $O(n)$

Triangualtions, complexity

Triangulation of P:
subdivision of $\mathrm{conv}(P)$ into triangles (simplices) whose vertex set is P

$P \subset \mathbb{R}^2 \Rightarrow$ triangulation has total complexity $O(n)$

"Good" triangulation?

- Terrain reconstruction: Avoid long skinny triangles

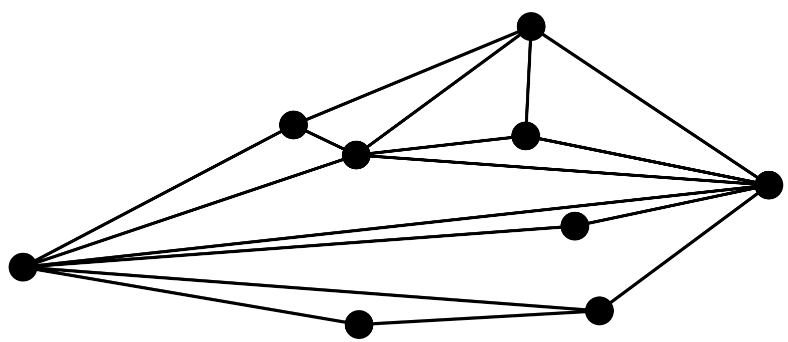
Triangualtions, complexity

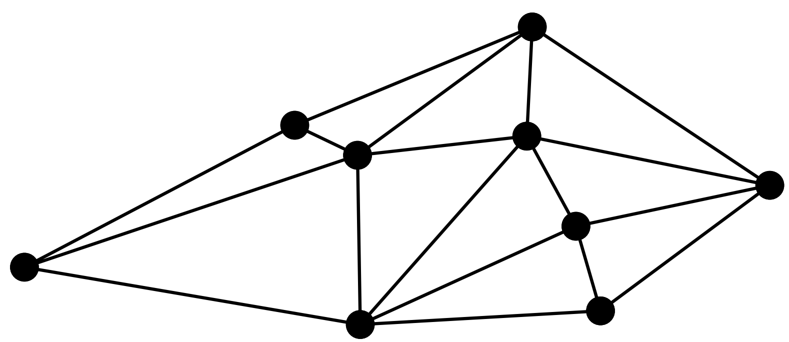
Triangulation of P:
subdivision of $\mathrm{conv}(P)$ into triangles (simplices) whose vertex set is P

$P \subset \mathbb{R}^2 \Rightarrow$ triangulation has total complexity $O(n)$

"Good" triangulation?

- Terrain reconstruction: Avoid long skinny triangles





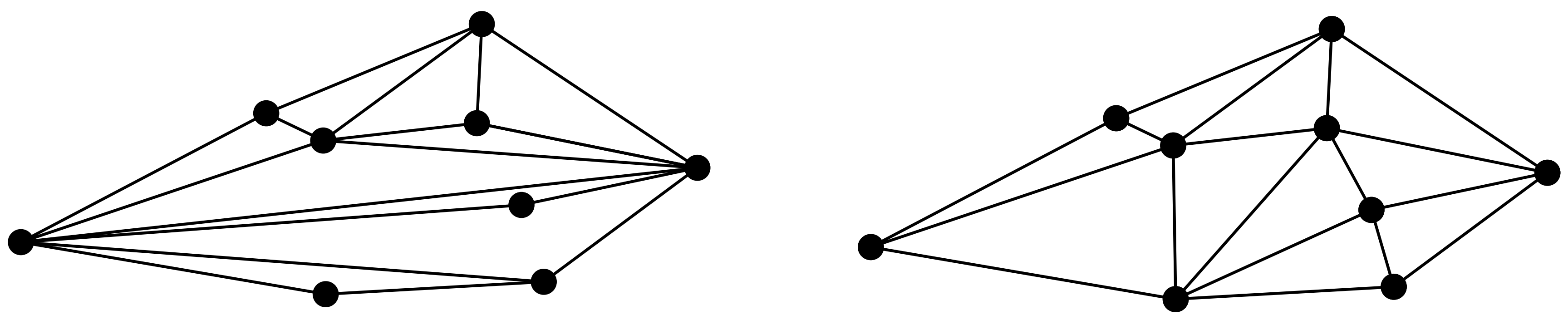
Triangualtions, complexity

Triangulation of P:
subdivision of $\mathrm{conv}(P)$ into triangles (simplices) whose vertex set is P

$P \subset \mathbb{R}^2 \Rightarrow$ triangulation has total complexity $O(n)$

"Good" triangulation?

- Terrain reconstruction: Avoid long skinny triangles



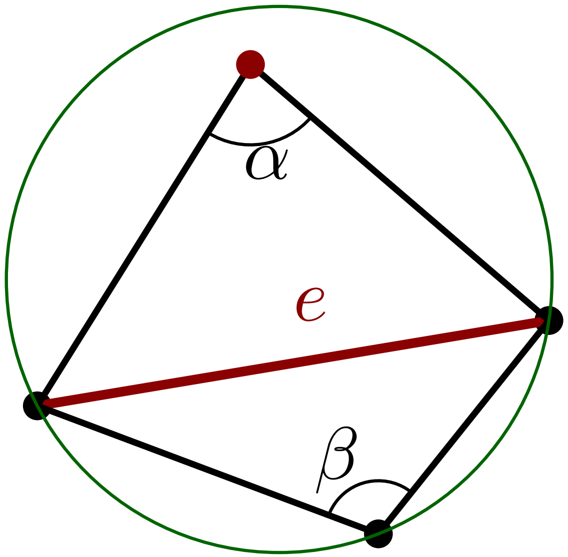
- Distance along triangualtion edges approximates Euclidean distance

Delaunay triangulation definition

Definition A Delaunay triangulation of P is a triangulation where the circumcircle of any triangle has no points of P in its interior.

Delaunay triangulation definition

Definition A Delaunay triangulation of P is a triangulation where the circumcircle of any triangle has no points of P in its interior.

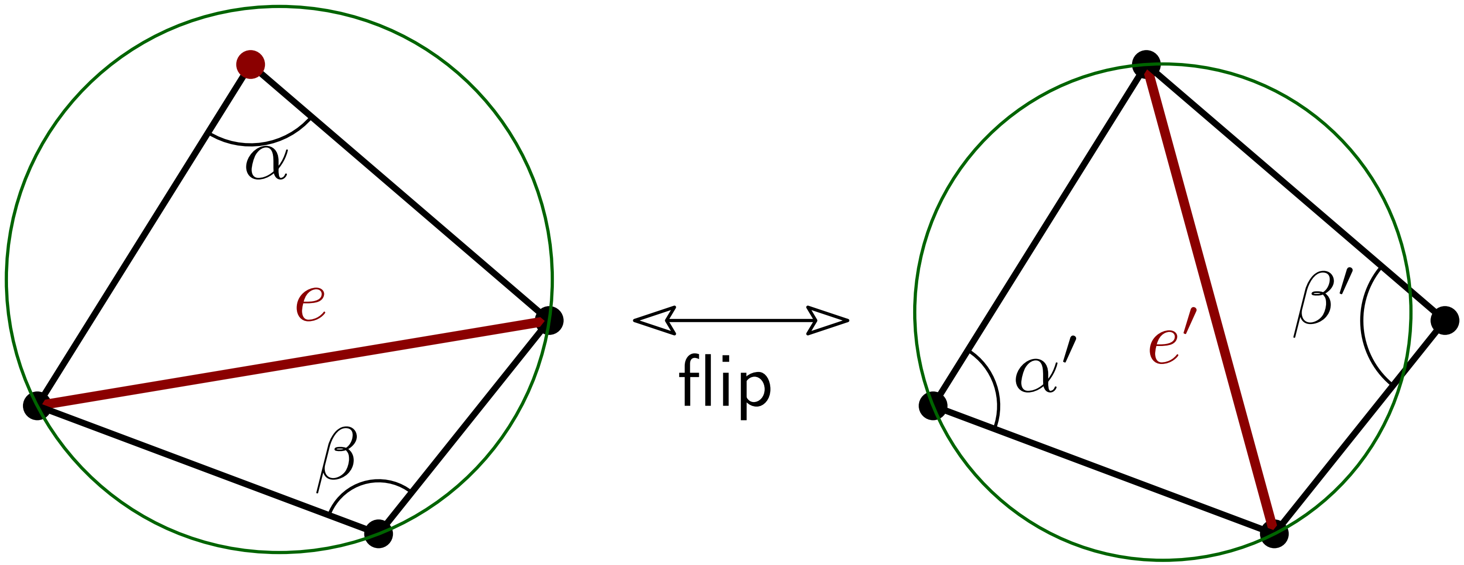


bad triangles

$\alpha + \beta > \pi$

Delaunay triangulation definition

Definition A Delaunay triangulation of P is a triangulation where the circumcircle of any triangle has no points of P in its interior.



bad triangles

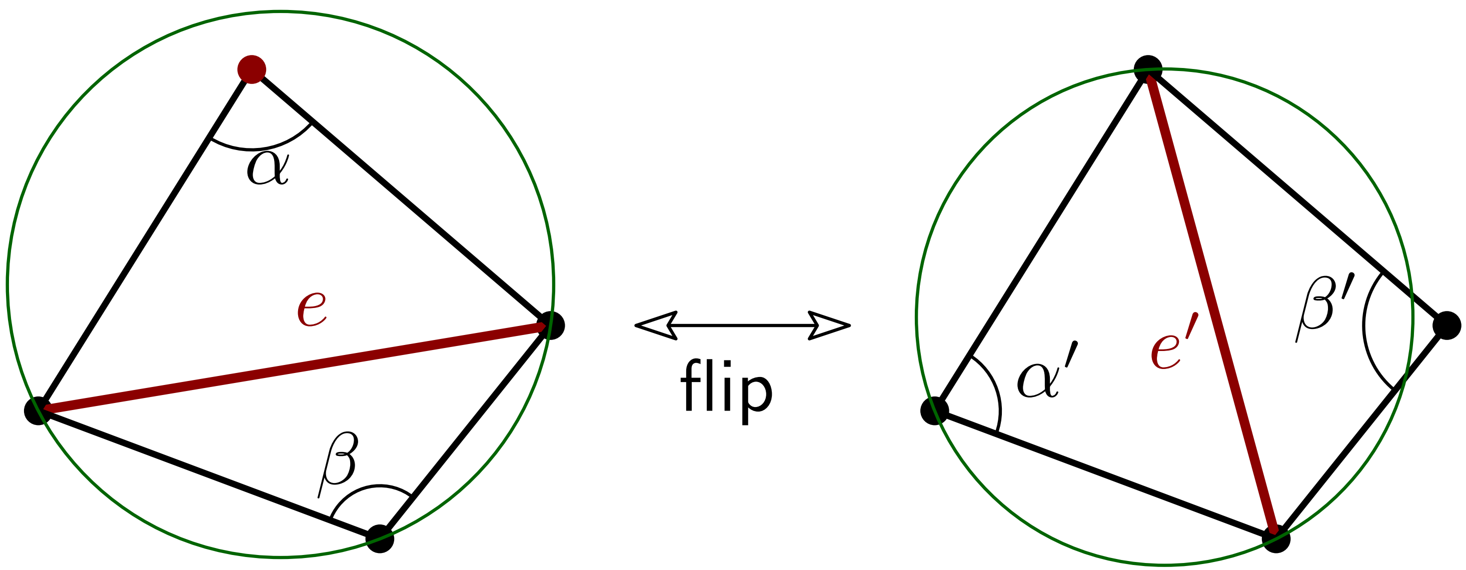
$\alpha + \beta > \pi$

good triangles

$\alpha' + \beta' \leq \pi$

Delaunay triangulation definition

Definition A Delaunay triangulation of P is a triangulation where the circumcircle of any triangle has no points of P in its interior.



bad triangles

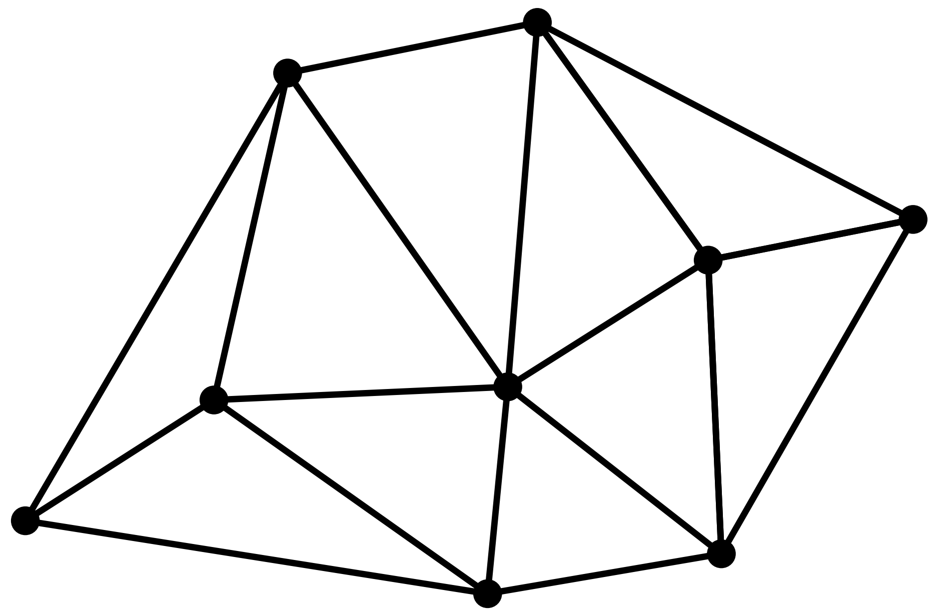
$\alpha + \beta > \pi$

good triangles

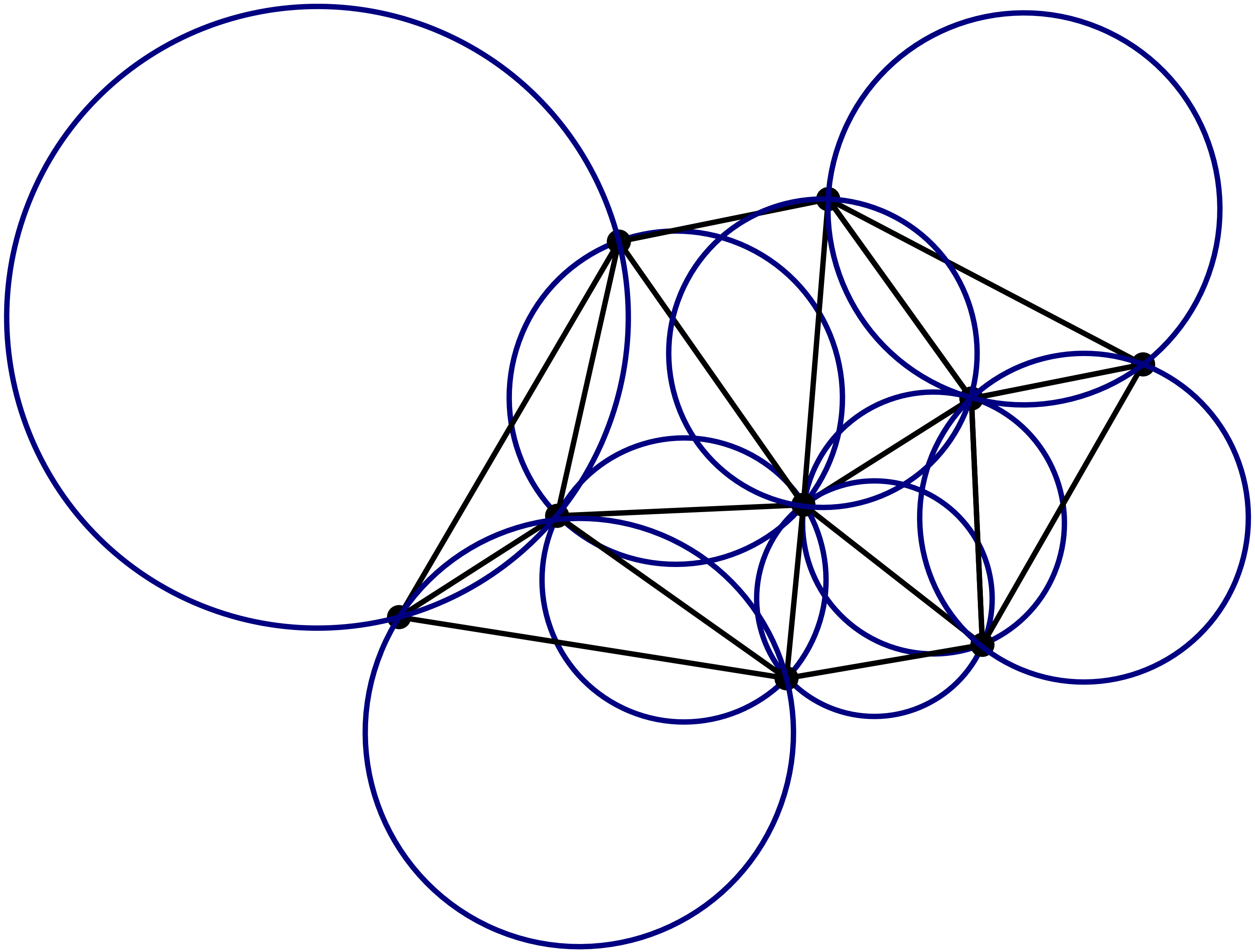
$\alpha' + \beta' \leq \pi$

DT is a triangulation whose angles (when ordered in increasing sequence) are lexicographically maximized.

Example



Example

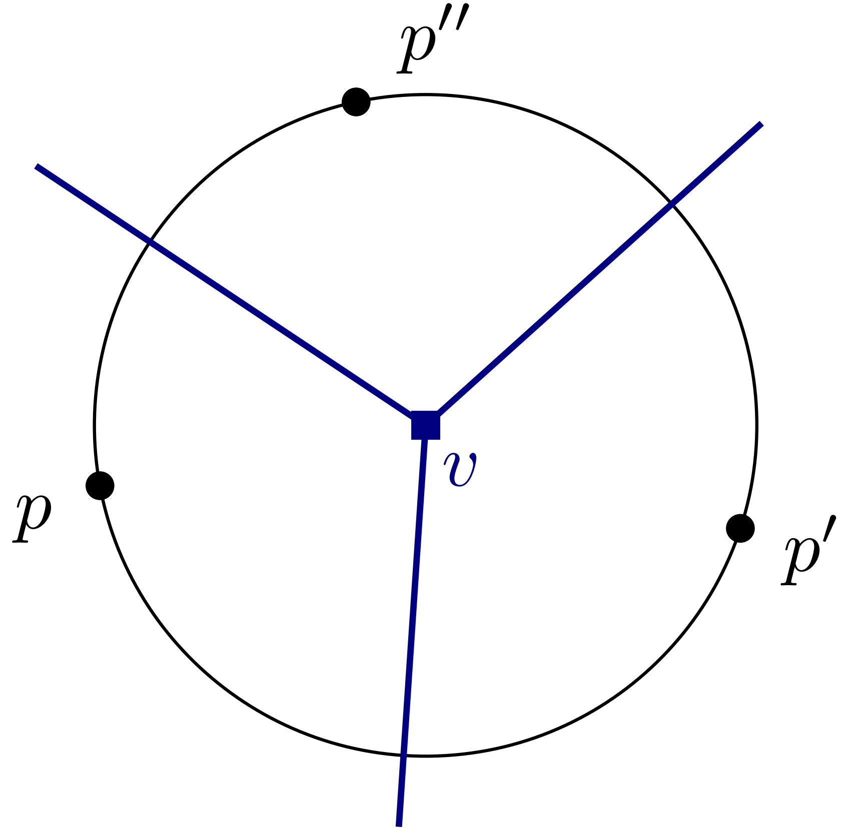


The dual of Voronoi

Voronoi vertex v at circumcenter of $pp'p''$

$\Updownarrow$

circumcircle of $pp'p''$ has no point of P in its interior

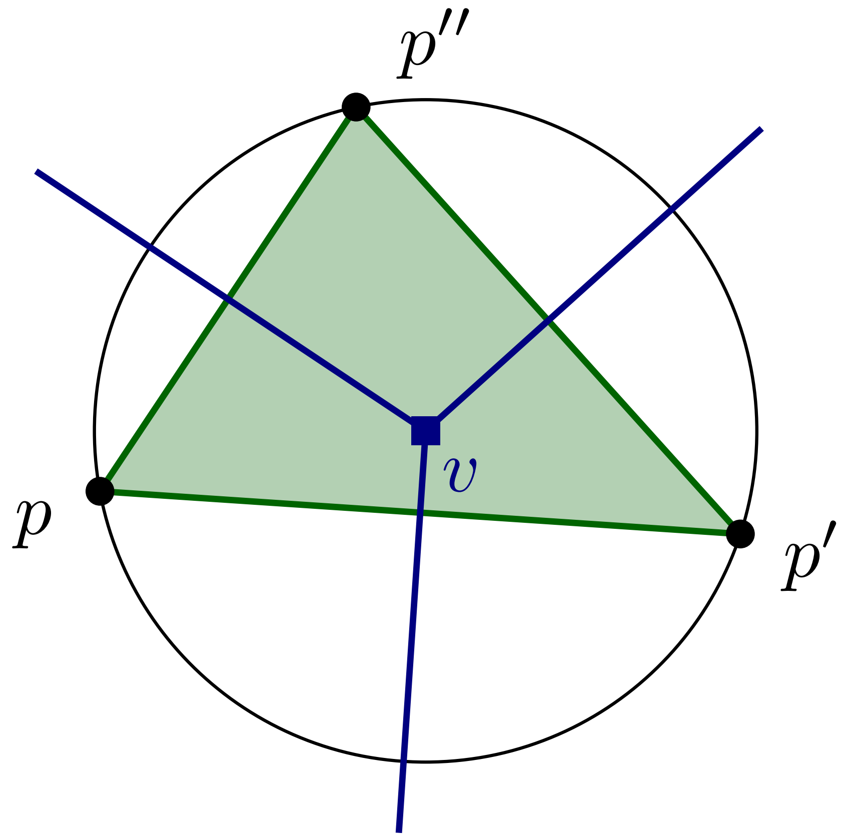


The dual of Voronoi

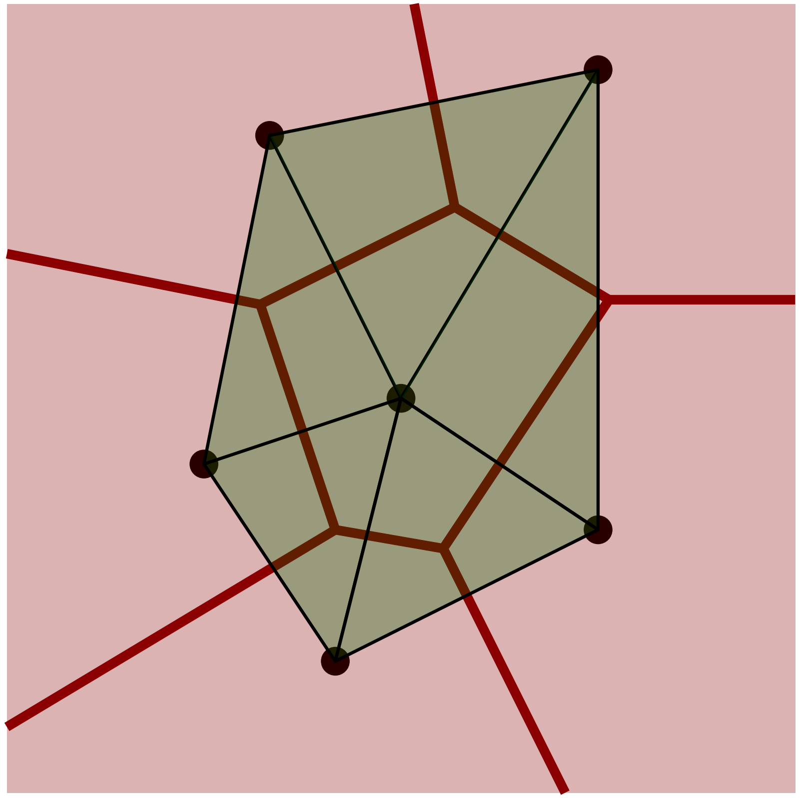
Voronoi vertex v at circumcenter of $pp'p''$

$pp'p''$ is a triangle in the Delaunay triangulation

circumcircle of $pp'p''$ has no point of P in its interior



Example: Voronoi and Delaunay

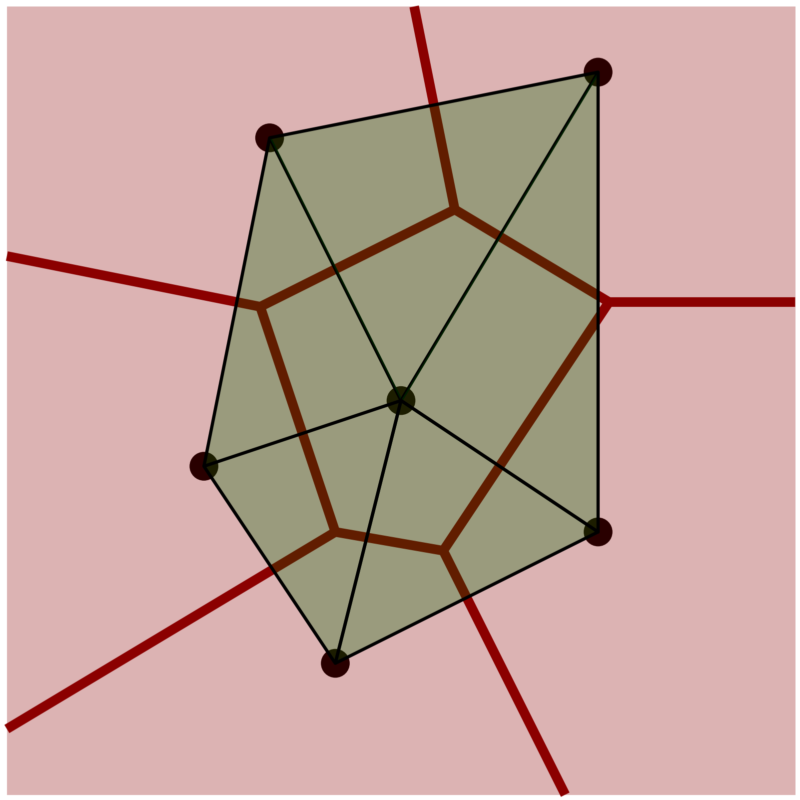


Voronoi edges

dual (Delaunay) edges

they define the Delaunay Graph

Example: Voronoi and Delaunay



Voronoi edges

dual (Delaunay) edges

they define the Delaunay Graph

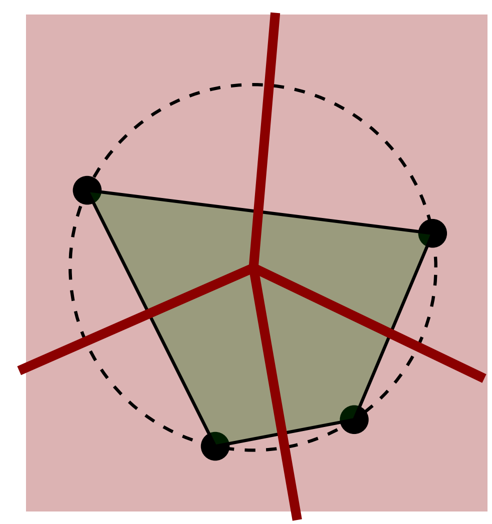
≥ 4 points on same circle $\Rightarrow$

Vor. vertex of degree ≥ 4

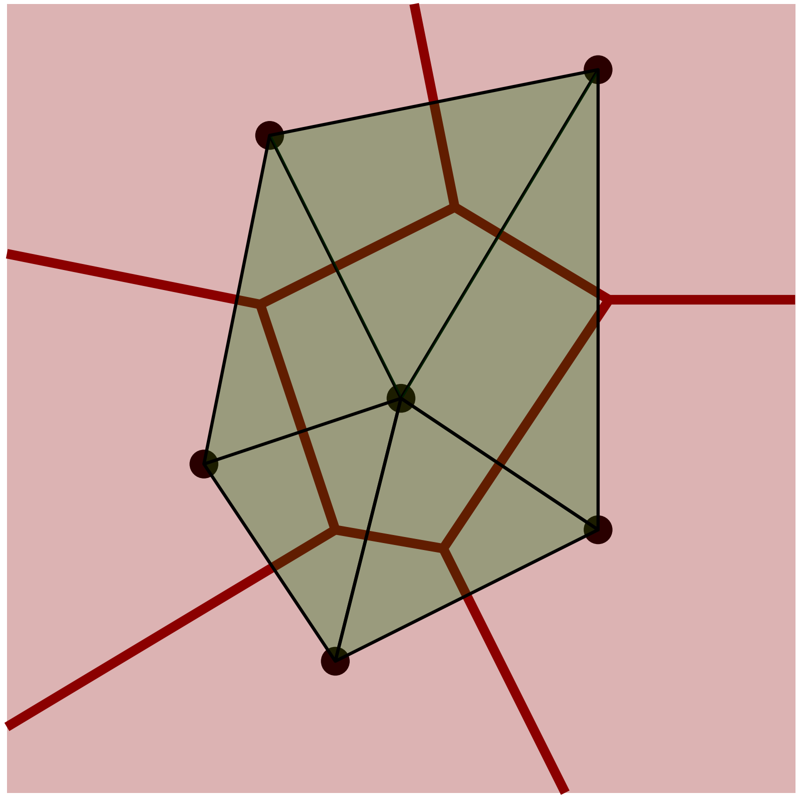
$\Updownarrow$

Face F of size ≥ 4 in Delaunay graph

(any triangulation of F has good triangles)



Example: Voronoi and Delaunay



Voronoi edges

dual (Delaunay) edges

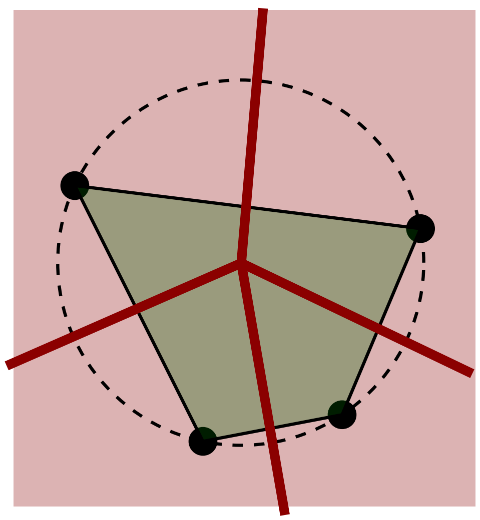
they define the Delaunay Graph

1) DG is plane graph
2) DT is unique and DT=DG
iff no 4 points on one circle

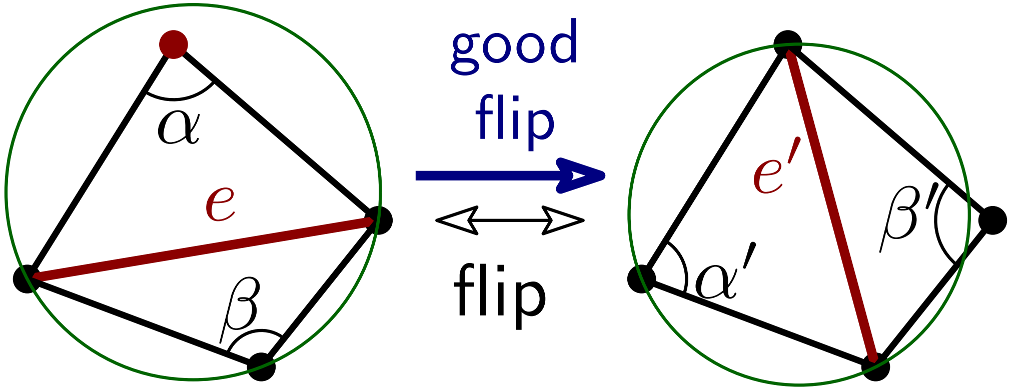
≥ 4 points on same circle $\Rightarrow$
Vor. vertex of degree ≥ 4

$\Updownarrow$

Face F of size ≥ 4 in Delaunay graph
(any triangulation of F has good triangles)



Incremental Delaunay with flips



bad triangles
$\alpha + \beta > \pi$

good triangles
$\alpha' + \beta' \leq \pi$

T is a Delaunay-tr.

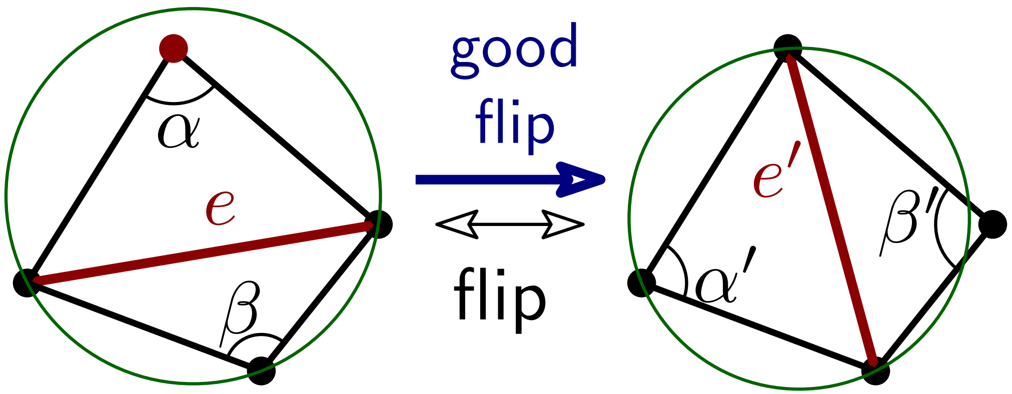
$\Leftrightarrow$

No bad triangles

$\Leftrightarrow$

No bad edges to flip

Incremental Delaunay with flips



bad triangles
$\alpha + \beta > \pi$

good triangles
$\alpha' + \beta' \leq \pi$

T is a Delaunay-tr.

$\Leftrightarrow$

No bad triangles

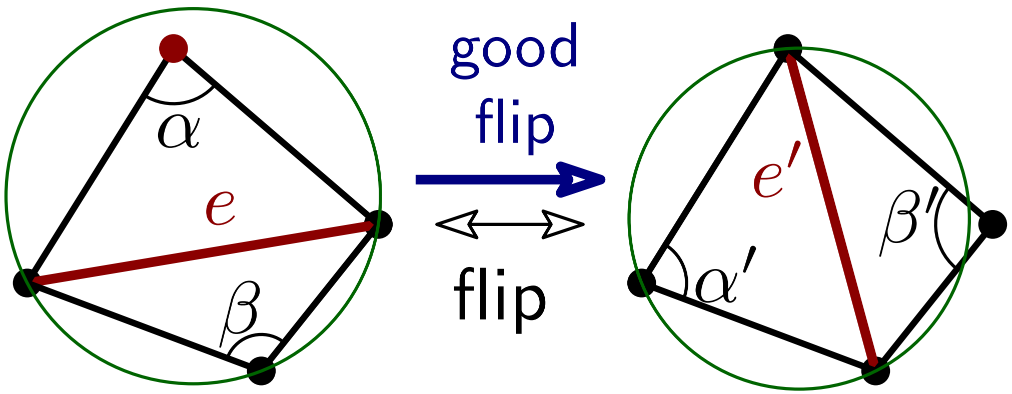
$\Leftrightarrow$

No bad edges to flip

Simple incremental algorithm:

- add points one at a time in random order
- maintain $DT(i) = DT(p_1 \ldots, p_i)$
- maintain special point location data structure on $DT(i)$

Incremental Delaunay with flips



bad triangles
$\alpha + \beta > \pi$

good triangles
$\alpha' + \beta' \leq \pi$

T is a Delaunay-tr.

$\Leftrightarrow$

No bad triangles

$\Leftrightarrow$

No bad edges to flip

Simple incremental algorithm:

- add points one at a time in random order
- maintain $DT(i) = DT(p_1 \ldots, p_i)$
- maintain special point location data structure on $DT(i)$

Use flips to update triangulation

Flip algorithm

After adding p_i:

1. Find triangle $\Delta(pp'p'') \in DT(i-1)$ where $p_i \in \Delta(pp'p'')$
2. Connect p_i to p, p', p'' (to get triangulation)
3. Flip until no more bad edges,
   updating point location throughout

Flip algorithm

After adding p_i:

1. Find triangle $\Delta(pp'p'') \in DT(i-1)$ where $p_i \in \Delta(pp'p'')$
2. Connect p_i to p, p', p'' (to get triangulation)
3. Flip until no more bad edges,
   updating point location throughout

Could be $\Omega(n)$ flips!

Flip algorithm

After adding p_i:

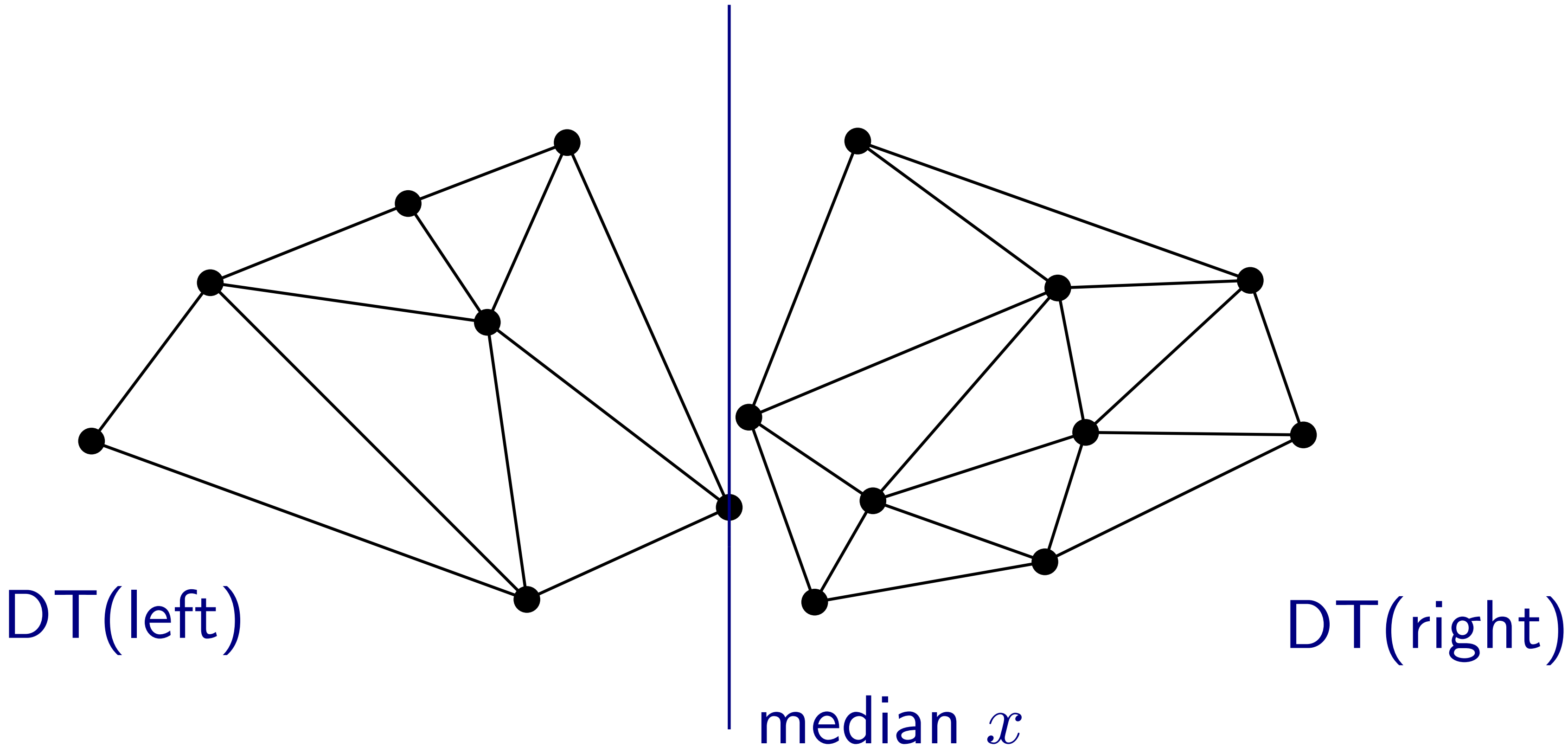
1. Find triangle $\Delta(pp'p'') \in DT(i-1)$ where $p_i \in \Delta(pp'p'')$
2. Connect p_i to p, p', p'' (to get triangulation)
3. Flip until no more bad edges,
   updating point location throughout

Could be $\Omega(n)$ flips!

Theorem The randomized incremental construction has expected running time $O(n \log n)$ and needs $O(n)$ space in expectation.

Delaunay triangulation via divdie and conquer (Guibas and Stolfi, 1985)

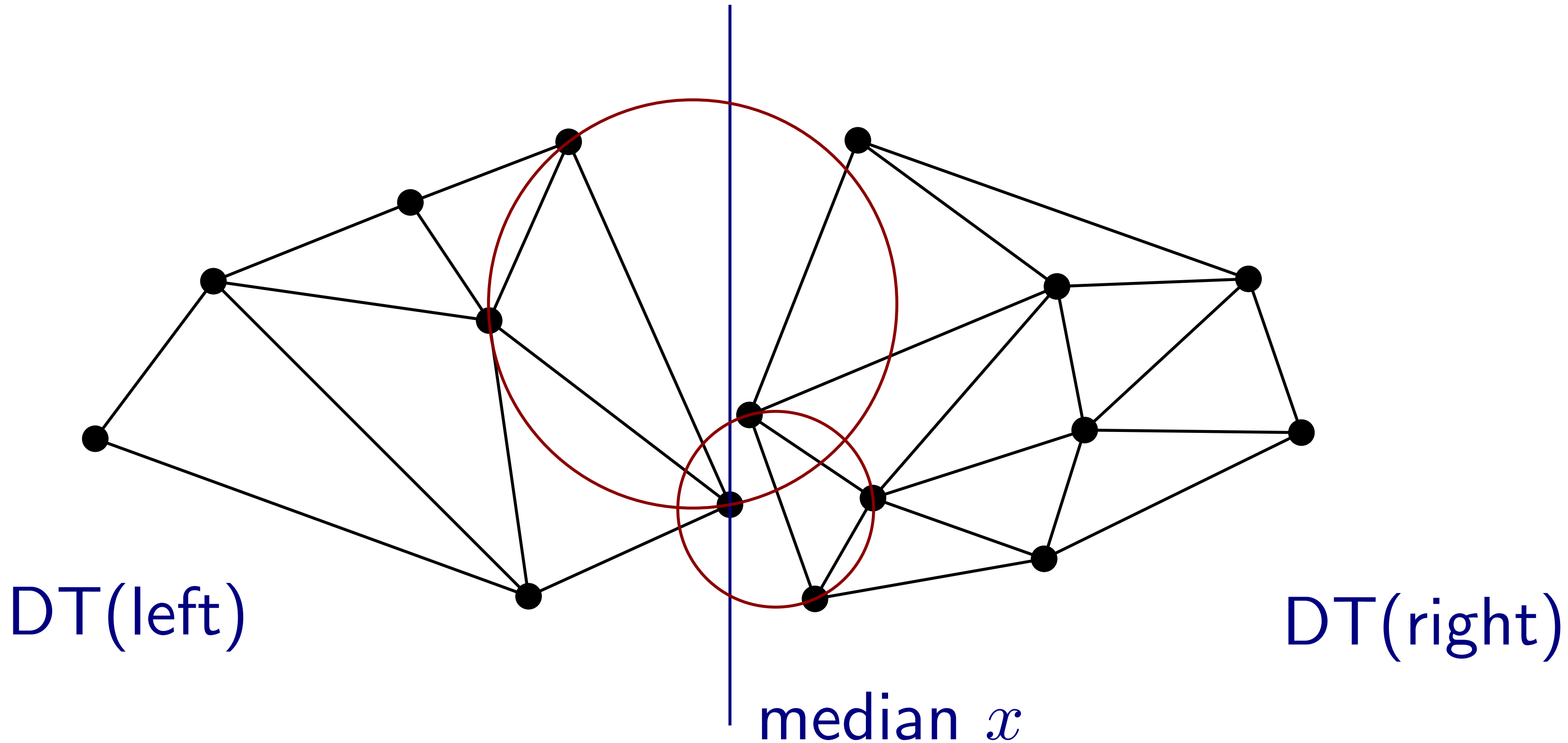
Divide and conquer DT



Task: merge in $O(n)$ time

$$T(n) = 2T(n/2) + O(n)$$

Divide and conquer DT

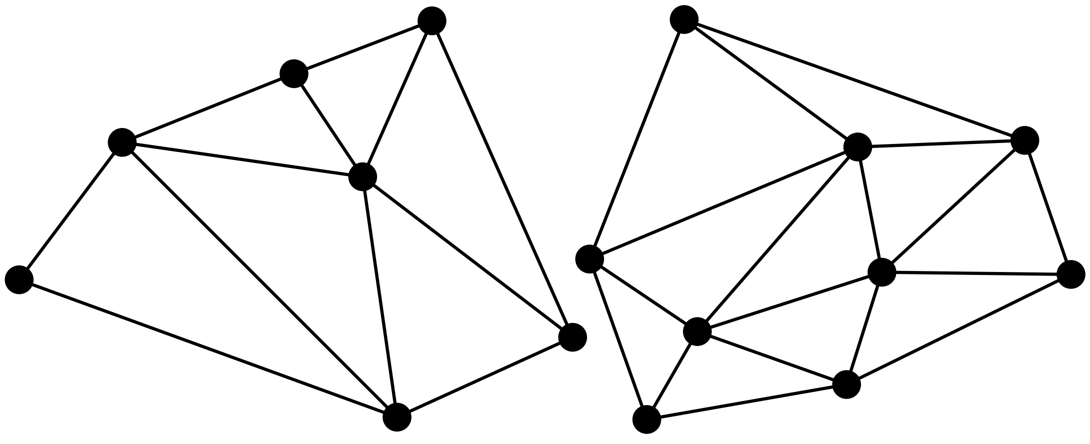


Task: merge in $O(n)$ time

$$T(n) = 2T(n/2) + O(n)$$

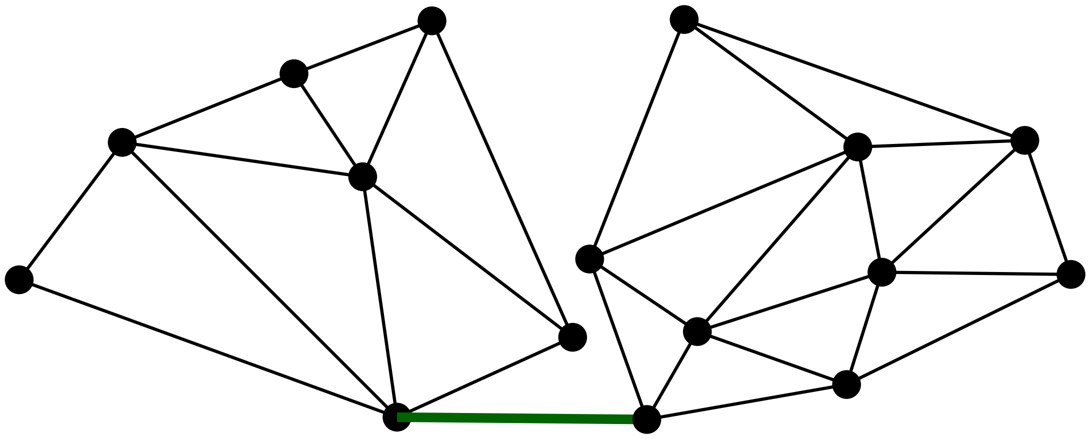
Some triangles became bad...

Bubble-up merge



Start with common lower tangent. $O(n)$

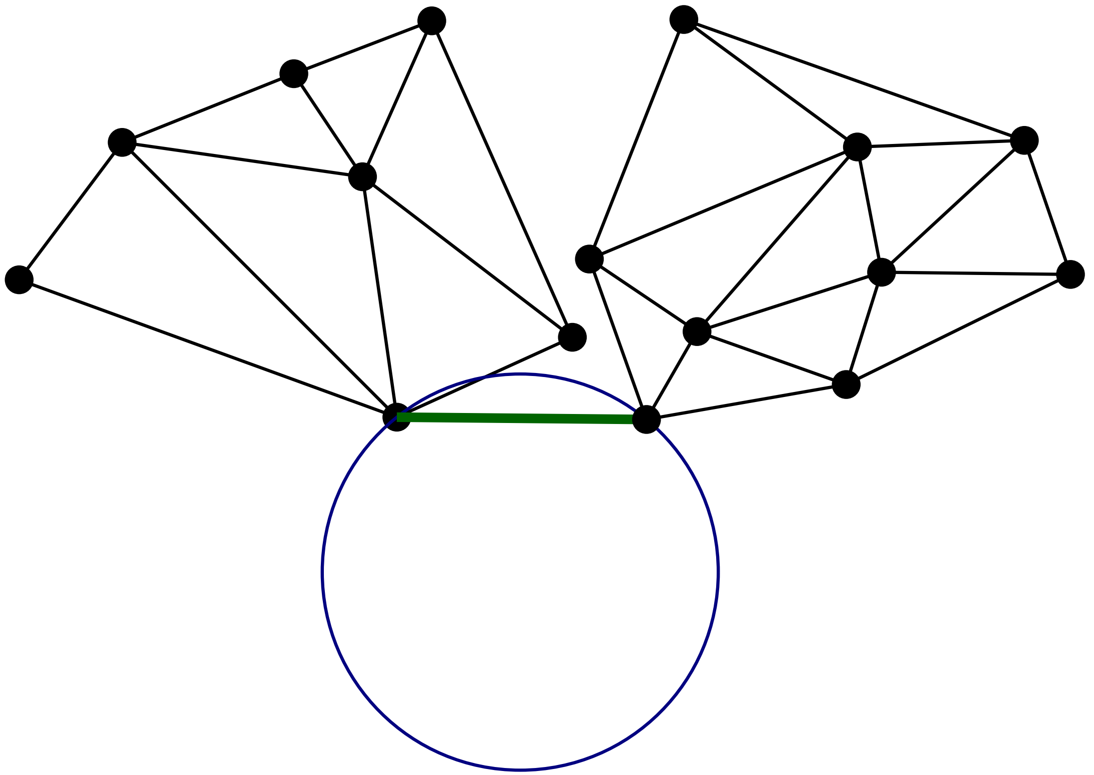
Bubble-up merge



Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

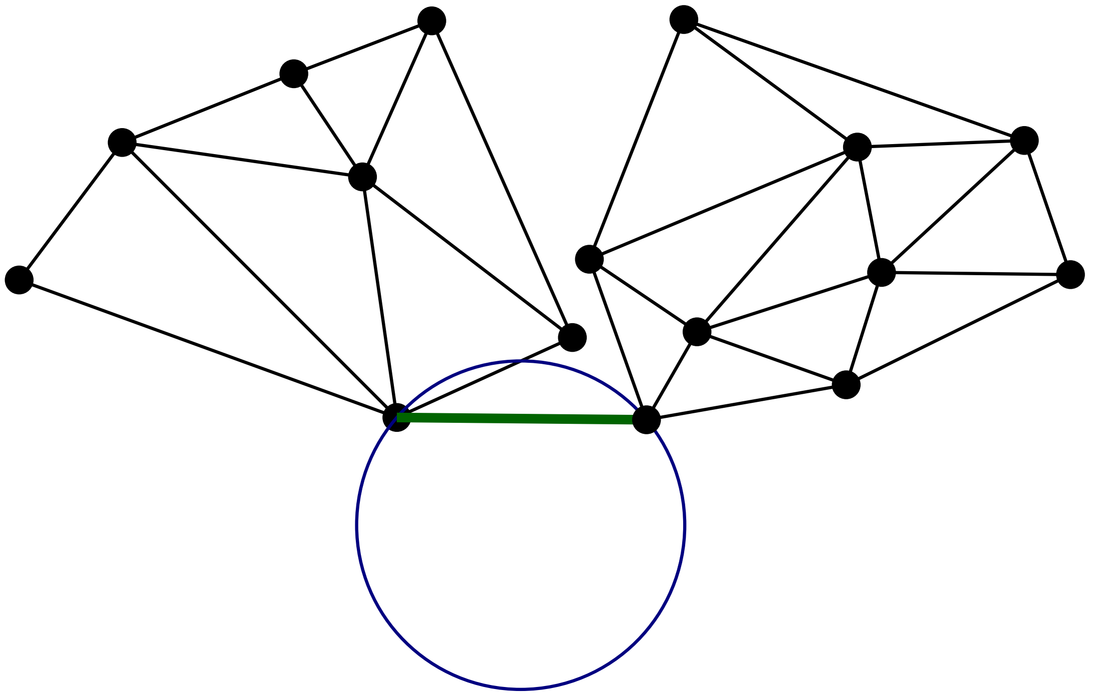
Bubble-up merge



Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

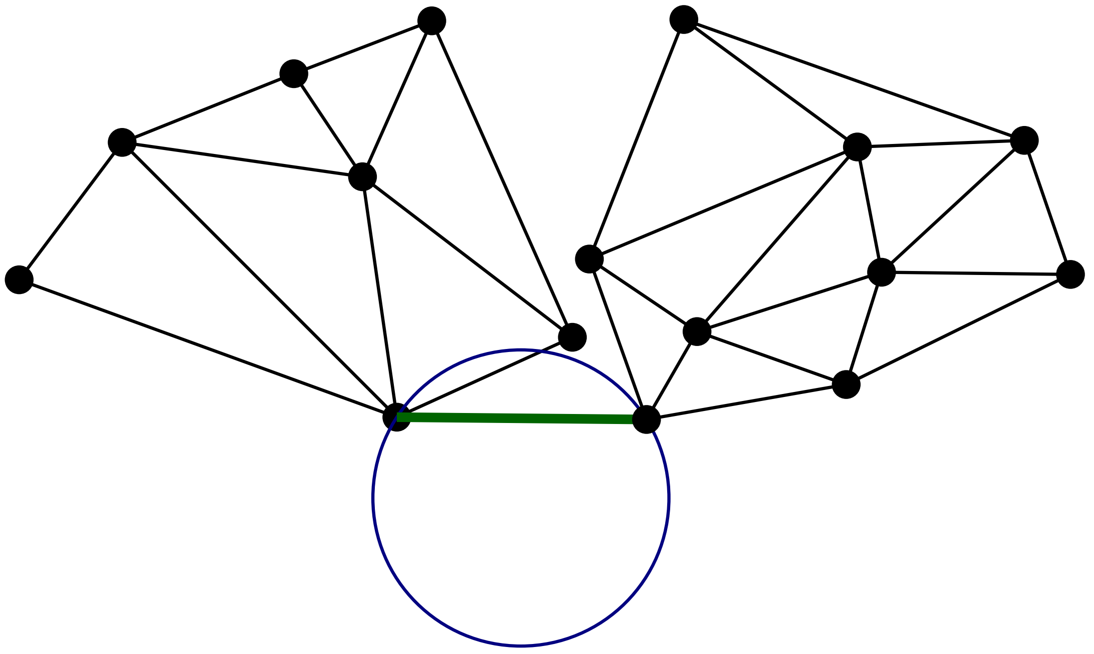
Bubble-up merge



Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

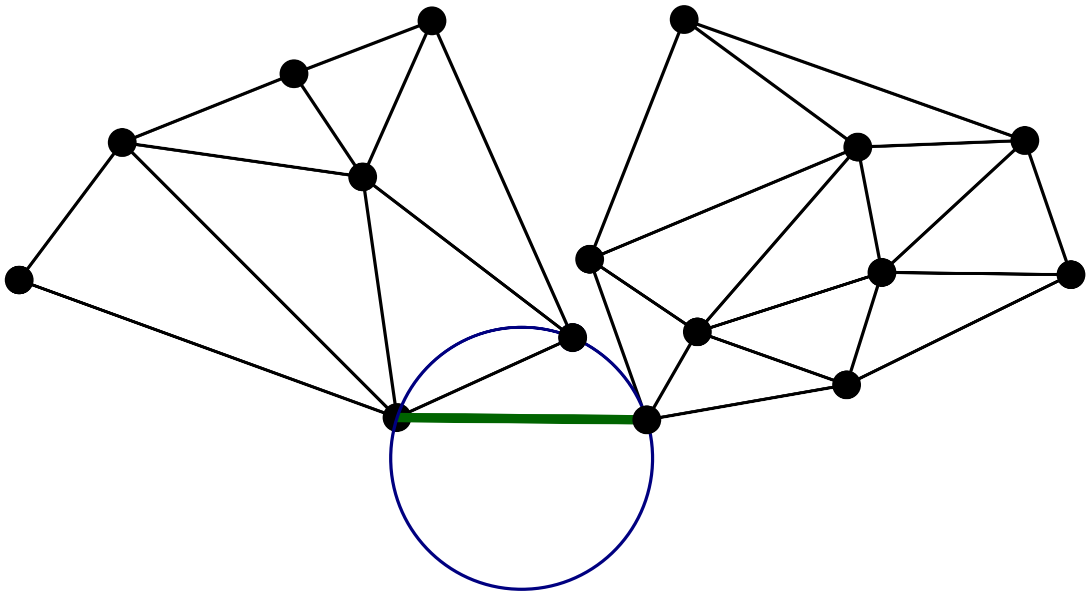
Bubble-up merge



Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

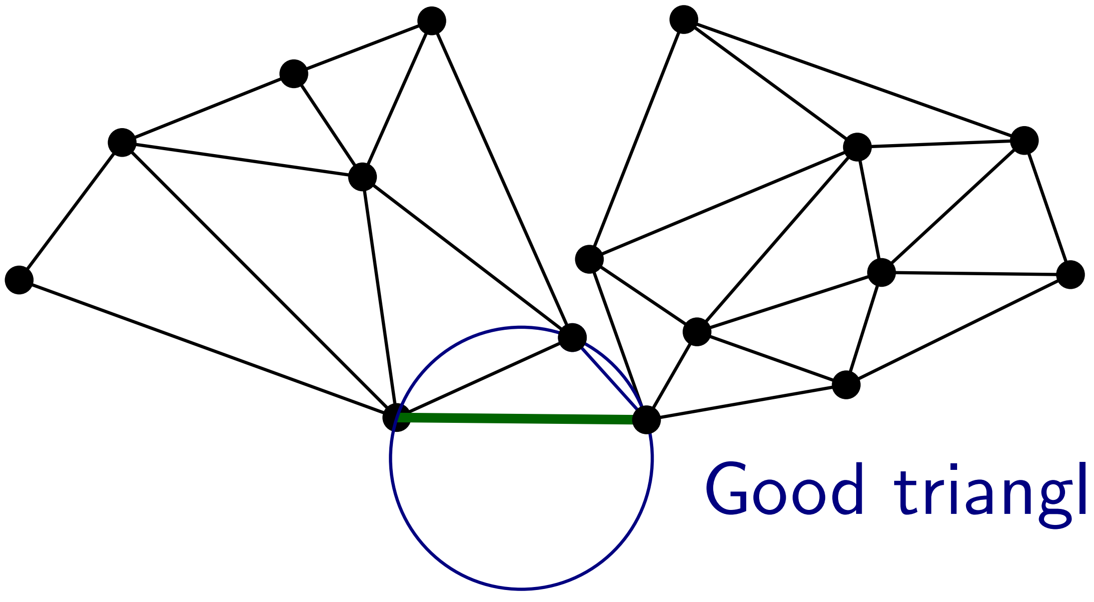
Bubble-up merge



Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

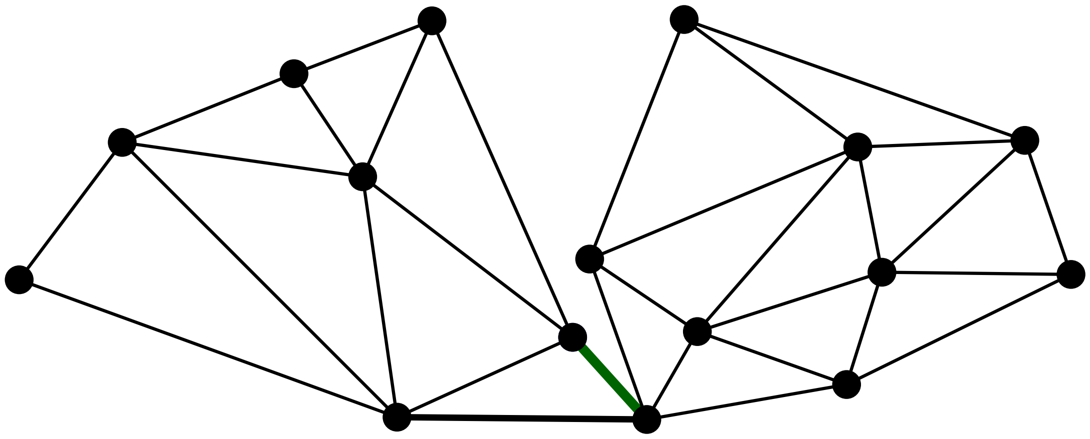
Bubble-up merge



Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

Bubble-up merge

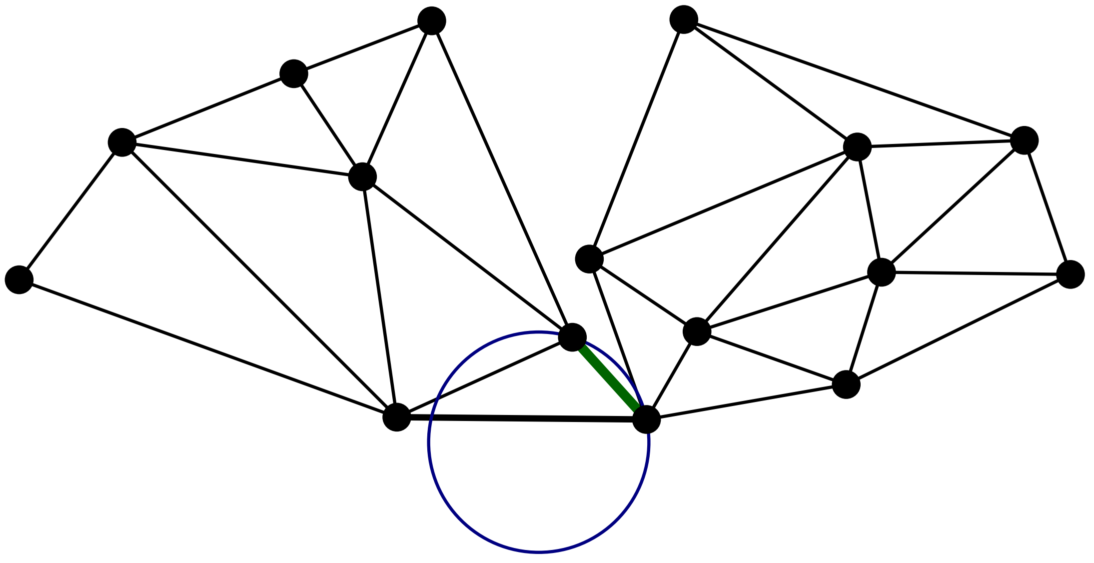


Good triangle! new edge found

Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

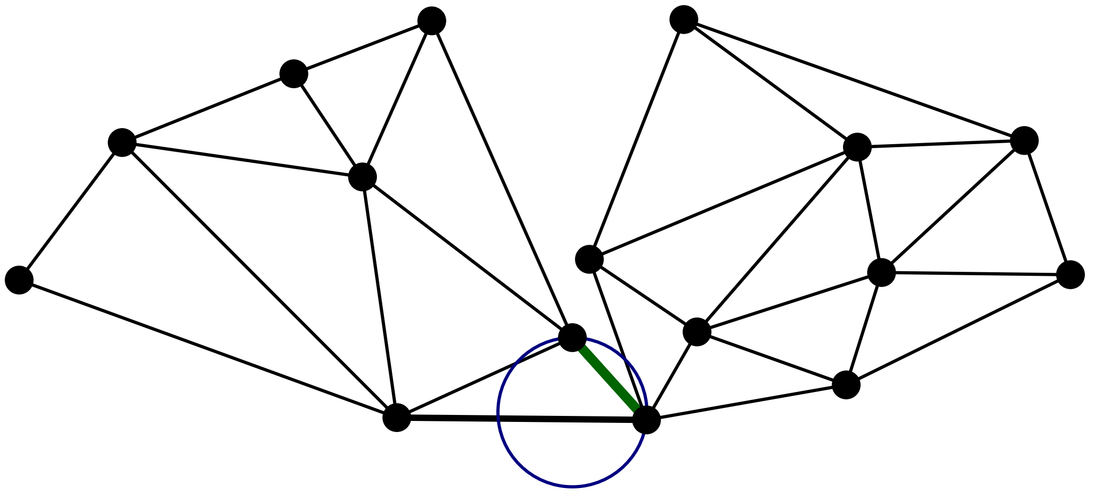
Bubble-up merge



Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

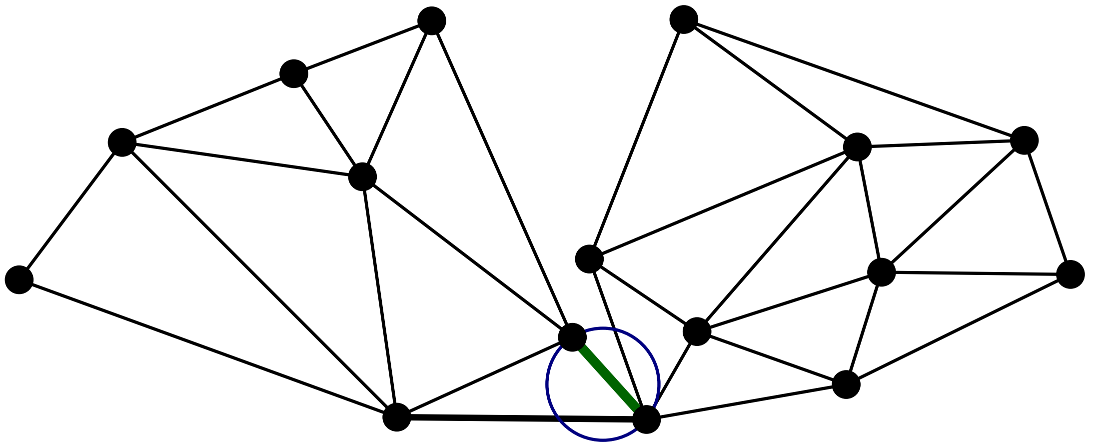
Bubble-up merge



Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

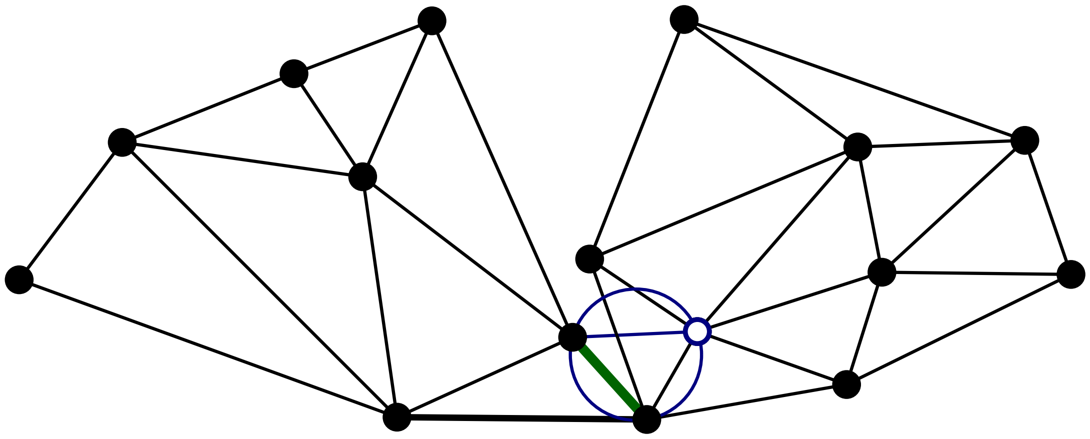
Bubble-up merge



Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

Bubble-up merge

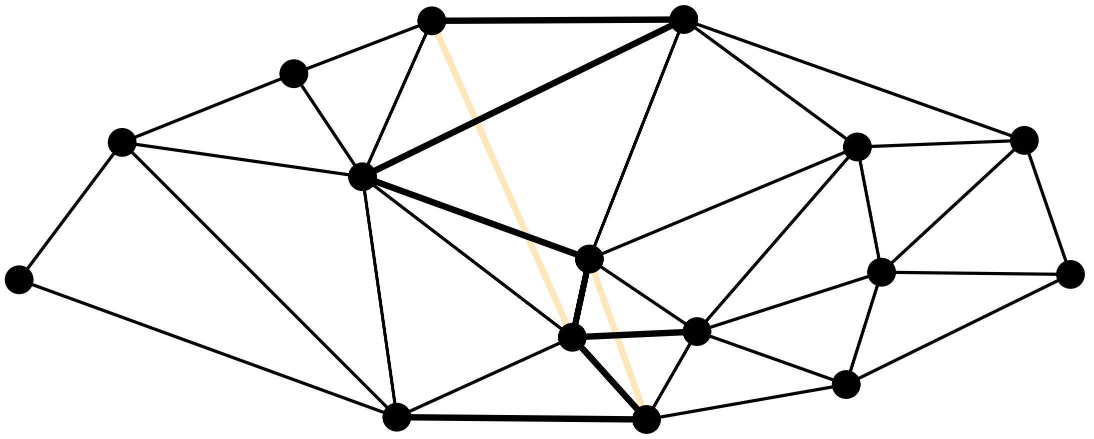


Good triangle! new edge found

Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

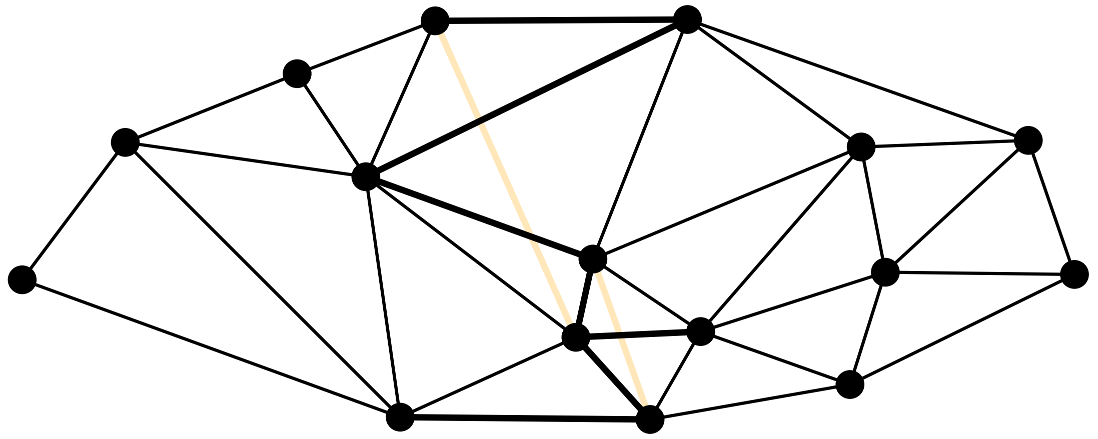
Bubble-up merge



Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

Bubble-up merge



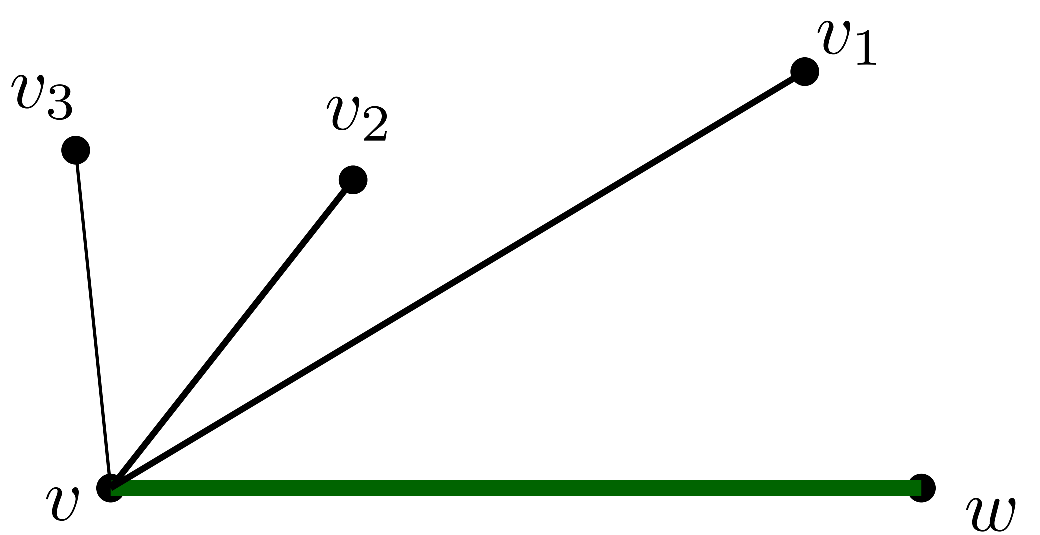
Start with common lower tangent. $O(n)$

Push a bubble through the base edge until new vertex is hit

Properties:

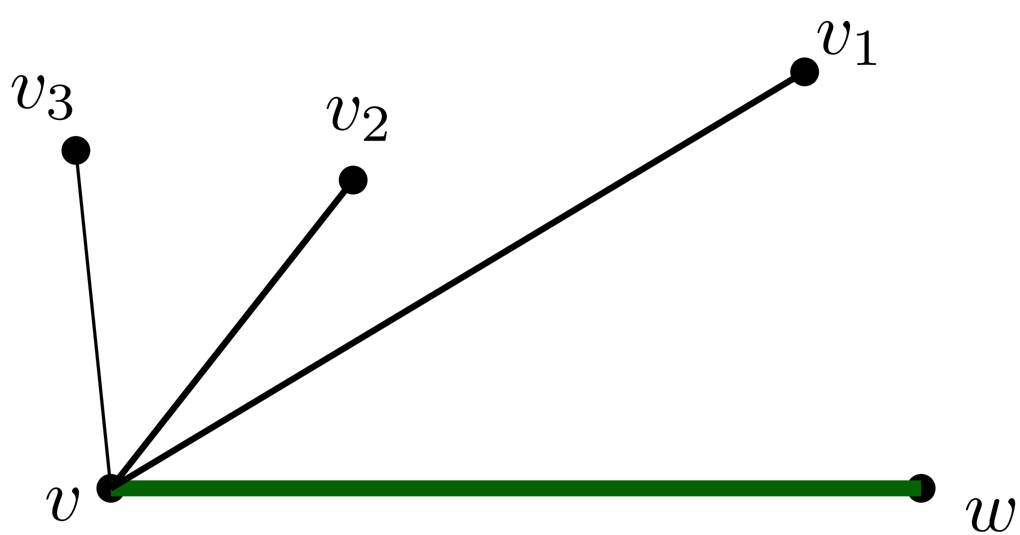
1. All bubbles are empty
2. edge deletions are justified (they intersect some valid edge)
3. gives triangulation with only valid triangles $\Rightarrow$ gives a DT

Finding the next bubble



New vertex is DT-neighbor of v or w. (Find candidates v', w' choose best)

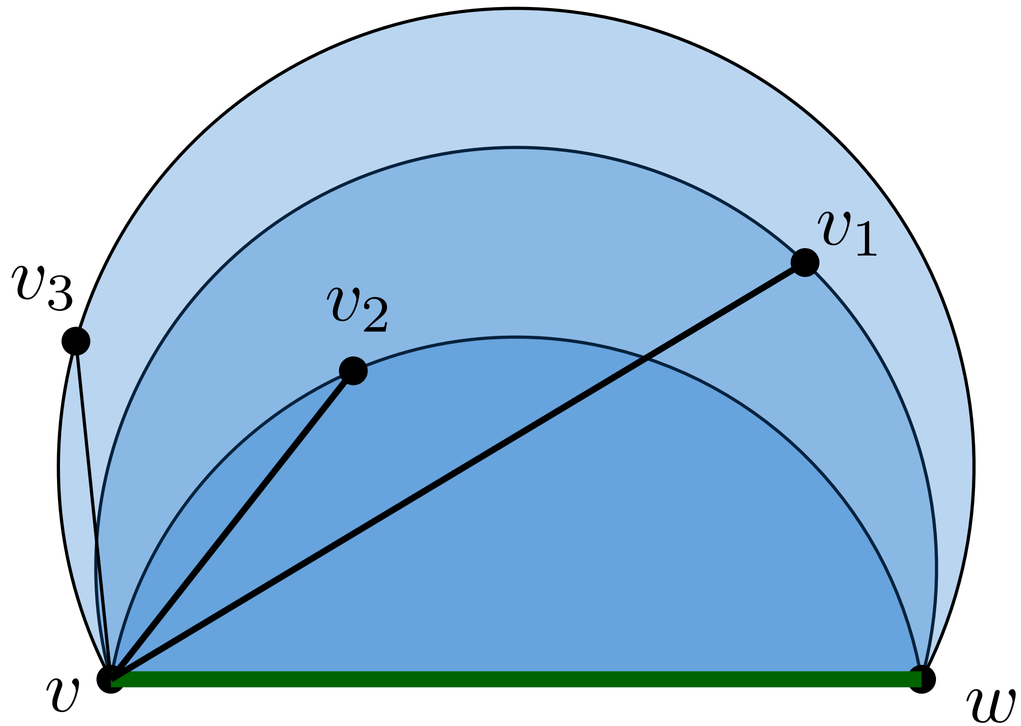
Finding the next bubble



New vertex is DT-neighbor of v or w. (Find candidates v', w' choose best)

$v_1, v_2, \ldots$: neighbors of v in CCW order after w

Finding the next bubble



New vertex is DT-neighbor of v or w. (Find candidates v', w' choose best)

$v_1, v_2, \ldots$: neighbors of v in CCW order after w

Claim There is an i such that

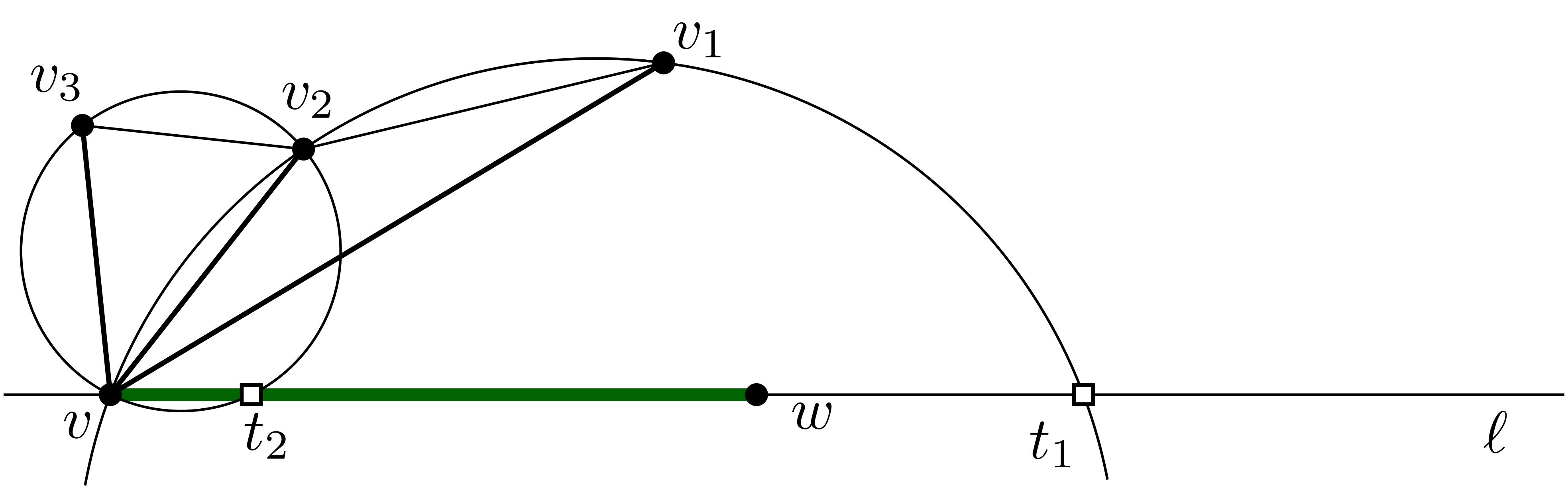
$$\cdots \supset slice(vv_{i-1}w) \supset slice(vv_iw) \subset slice(vv_{i+1}w) \subset \ldots$$

Proof of unimodality

Claim There is an i such that

$$\cdots \supset slice(vv_{i-1}w) \supset slice(vv_iw) \subset slice(vv_{i+1}w) \subset \ldots$$

t_i := other intersection of ℓ and $circle(vv_iv_{i+1})$



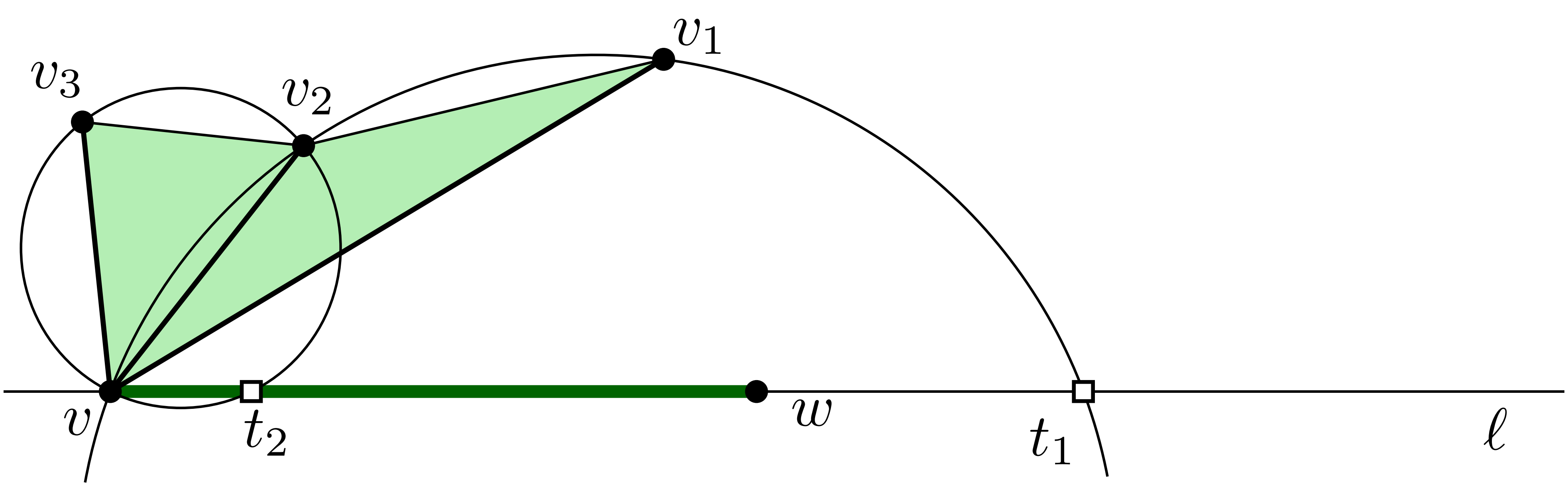
Proof of unimodality

Claim There is an i such that

$$\cdots \supset slice(vv_{i-1}w) \supset slice(vv_iw) \subset slice(vv_{i+1}w) \subset \ldots$$

$t_i :=$ other intersection of ℓ and $circle(vv_iv_{i+1})$

$\Delta(vv_{i-1}v_i), \Delta(vv_iv_{i+1})$ are empty triangles in DT(left)
$\Rightarrow t_1, t_2, \ldots$ moves left on ℓ



Proof of unimodality

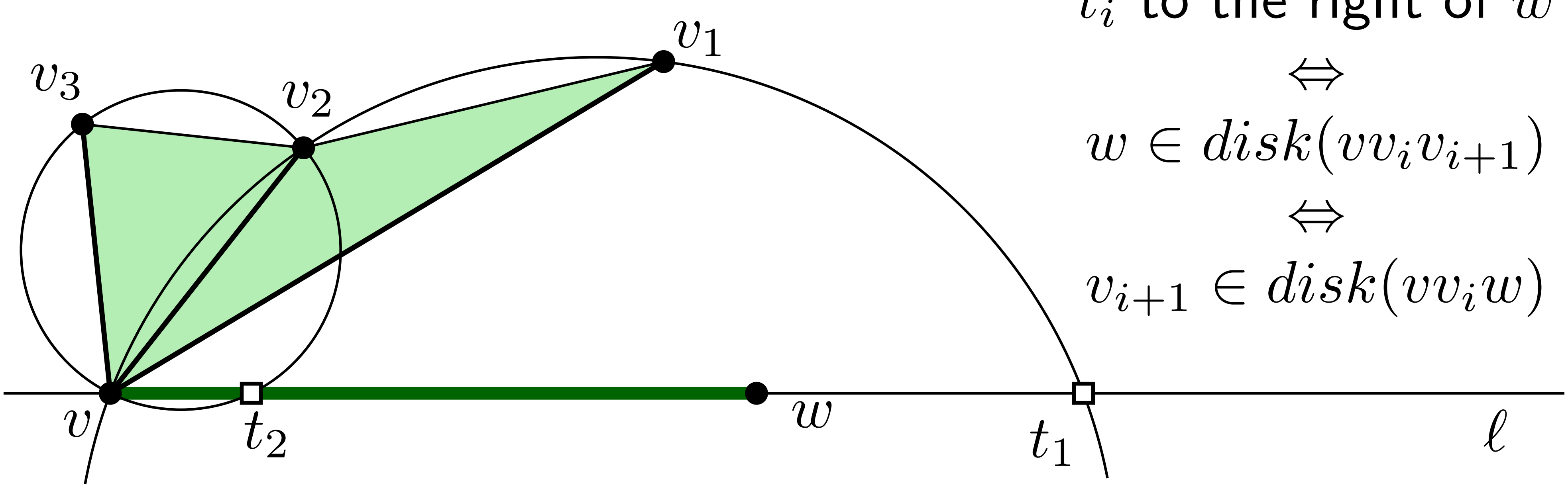
Claim There is an i such that

$$\cdots \supset slice(vv_{i-1}w) \supset slice(vv_iw) \subset slice(vv_{i+1}w) \subset \ldots$$

$t_i :=$ other intersection of ℓ and $circle(vv_iv_{i+1})$

$\Delta(vv_{i-1}v_i), \Delta(vv_iv_{i+1})$ are empty triangles in DT(left)
$\Rightarrow t_1, t_2, \ldots$ moves left on ℓ

t_i to the right of w
$\Leftrightarrow$
$w \in disk(vv_iv_{i+1})$
$\Leftrightarrow$
$v_{i+1} \in disk(vv_iw)$



Proof of unimodality

Claim There is an i such that

$$\dots \supset slice(vv_{i-1}w) \supset slice(vv_iw) \subset slice(vv_{i+1}w) \subset \dots$$

$t_i :=$ other intersection of ℓ and $circle(vv_iv_{i+1})$

$\Delta(vv_{i-1}v_i), \Delta(vv_iv_{i+1})$ are empty triangles in DT(left)

$\Rightarrow t_1, t_2, \dots$ moves left on ℓ

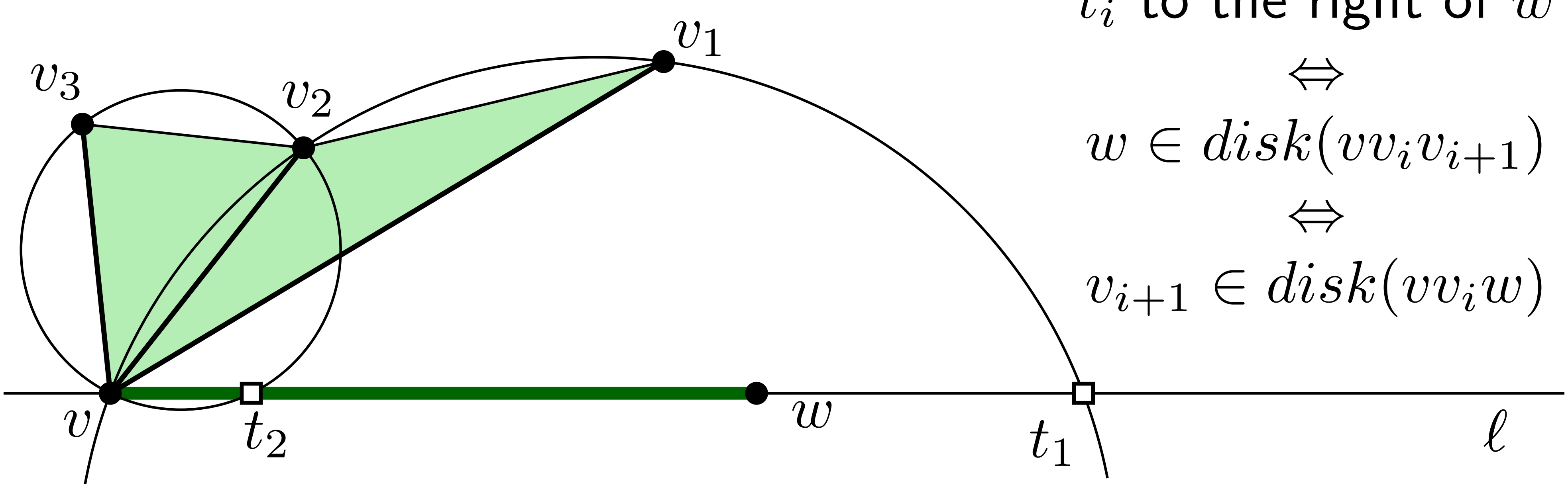
t_i to the right of w

$\Leftrightarrow$

$w \in disk(vv_iv_{i+1})$

$\Leftrightarrow$

$v_{i+1} \in disk(vv_iw)$



Next hit in DTleft: v_i where t_i is first to the left of w

Bubble-up merge running time

Any edge vv_i passed is not DT edge (not empty disk)
$\Rightarrow$ delete such edges

Bubble-up merge running time

Any edge vv_i passed is not DT edge (not empty disk)
$\Rightarrow$ delete such edges

- find common tangents
- starting at bottom tangent$= vw$:
  - find $v_i =$ next hit on left by stepping through $N(v)$ in CCW order, deleting passed edges
  - find $w_j =$ next hit on right by stepping through $N(w)$ in CW order, deleting passed edges
  - check which of v_i,w_j works
  - set vw as new edge

  until vw is other tangent

Bubble-up merge running time

Any edge vv_i passed is not DT edge (not empty disk)
$\Rightarrow$ delete such edges

- find common tangents $O(n)$
- starting at bottom tangent$= vw$:
  - find $v_i =$ next hit on left by stepping through $N(v)$ in CCW order, deleting passed edges
  - find $w_j =$ next hit on right by stepping through $N(w)$ in CW order, deleting passed edges
  - check which of v_i,w_j works
  - set vw as new edge

  until vw is other tangent

$O(1)$ steps per deleted edge, O(n) deleted edges

Bubble-up merge running time

Any edge vv_i passed is not DT edge (not empty disk)
$\Rightarrow$ delete such edges

- find common tangents $O(n)$
- starting at bottom tangent$= vw$:
  - find $v_i =$ next hit on left by stepping through $N(v)$ in CCW order, deleting passed edges
  - find $w_j =$ next hit on right by stepping through $N(w)$ in CW order, deleting passed edges
  - check which of v_i,w_j works
  - set vw as new edge

  until vw is other tangent

$O(1)$ steps per deleted edge, O(n) deleted edges

Bubble merge runs in $O(n)$